TURING 图灵程序设计丛书

no starch press

Learn Java the Easy Way

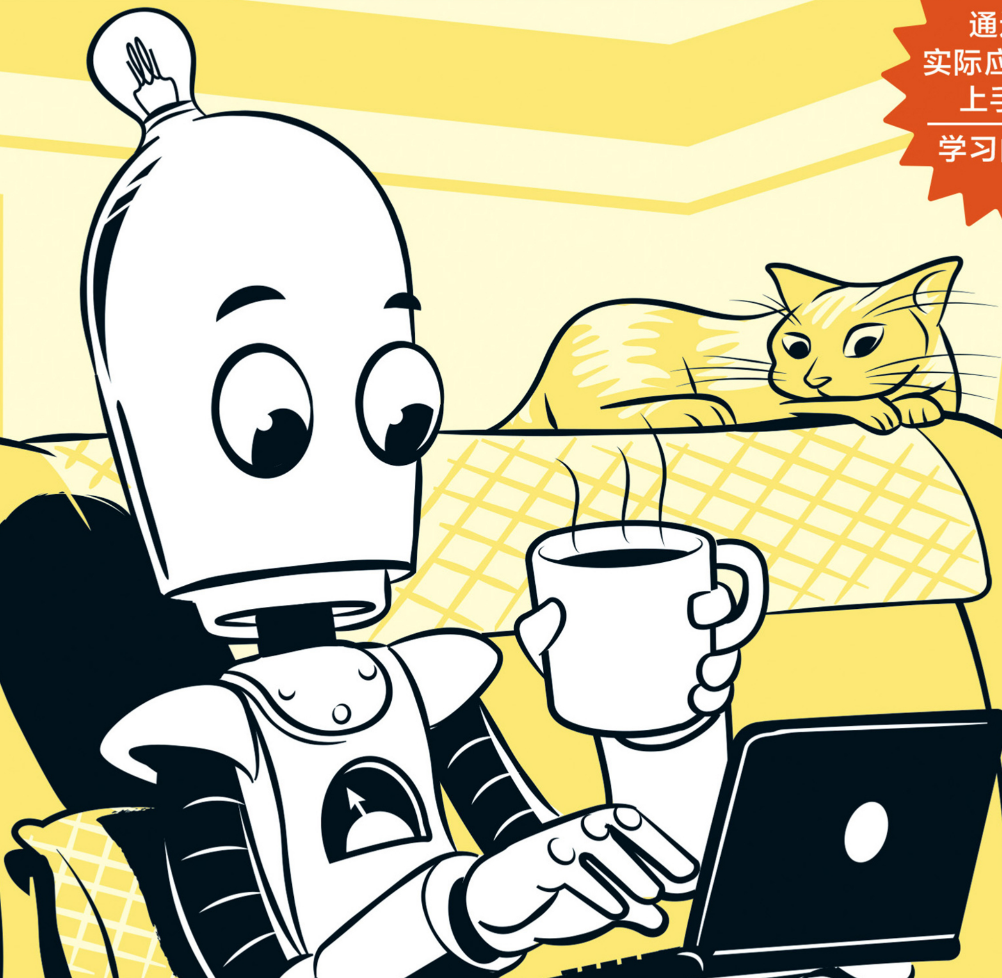
A Hands-On Introduction to Programming

Java轻松学

[美] Bryson Payne 著 袁国忠 译

通过开发
实际应用和游戏
上手Java

学习曲线平缓



中国工信出版集团 人民邮电出版社
POSTS & TELECOM PRESS

Bryson Payne

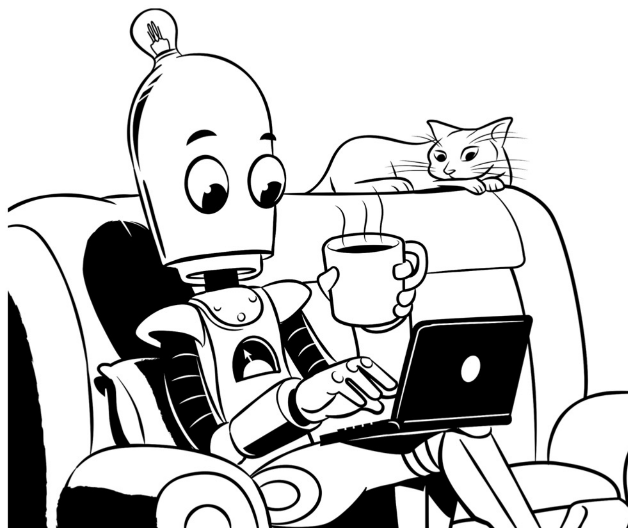
北佐治亚大学计算机科学系终身教授，并曾任该校CEO。从事Java教学工作近20年，深谙教学之道。一直致力于与全球的K-12学校合作以促进计算机科学教育，在Udemy上所开设的安全方面的培训课程吸引了全球150多个国家的数万名学生。

数字版权声明

TURING 图灵程序设计丛书

Java轻松学

[美] Bryson Payne 著　袁国忠 译



Learn Java the Easy Way

A Hands-On Introduction to Programming

人 民 邮 电 出 版 社
北　京

图书在版编目（C I P）数据

Java轻松学 / (美) 布赖森•佩恩 (Bryson Payne)
著 ; 袁国忠译. -- 北京 : 人民邮电出版社, 2018.5
(图灵程序设计丛书)
ISBN 978-7-115-48219-8

Ⅰ. ①J… Ⅱ. ①布… ②袁… Ⅲ. ①JAVA语言－程序
设计 Ⅳ. ①TP312.8

中国版本图书馆CIP数据核字(2018)第064321号

内 容 提 要

本书是 Java 基础教程类图书，通过开发实际的桌面和移动应用，从实战角度指导读者快速上手 Java 编程。主要内容包括：Java、Eclipse 和 Android Studio 的安装与设置，JShell 的用法，条件、循环、方法变量、类等 Java 编程概念，函数创建，GUI 构建，代码调试，常见错误的规避。

本书适合所有对 Java 编程感兴趣的初学者。

◆ 著　[美] Bryson Payne
译　袁国忠
责任编辑　岳新欣
执行编辑　李　敏
责任印制　周昇亮

◆ 人民邮电出版社出版发行　北京市丰台区成寿寺路11号
邮编　100164　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京　印刷

◆ 开本：800×1000　1/16
印张：16
字数：378千字　2018年 5 月第 1 版
印数：1 – 3 500册　2018年 5 月北京第 1 次印刷

著作权合同登记号　图字：01-2017-8627号

定价：59.00元

读者服务热线：(010)51095186转600　印装质量热线：(010)81055316

反盗版热线：(010)81055315

广告经营许可证：京东工商广登字 20170147 号

谨以此书献给妈妈，感谢您始终对我信任有加。

版 权 声 明

前　　言

Java被用于全球各地的数十亿台设备中。从移动应用到桌面软件，从最大的企业设备到最小的个人设备，它们背后都是 Java 在提供动力。学生、IT 专业人士以及任何有志于从事编程事业的人都将发现，Java 是必须学会的。作为从事了将近 20 年 Java 教学的计算机科学教授，我编写本书旨在帮助你像我学习编程时一样，通过开发实际项目来学习 Java。我发现，开发有趣、引人入胜且值得分享的实际应用和游戏，是学习编程的最佳方式。在本书中，你将创建简单的猜数游戏、用于与朋友交换消息的 Secret Messages 应用以及交互式绘图应用 BubbleDraw。你不需要有任何编程经验，但如果你学习过其他语言，也可通过这种实践方法更快地掌握 Java。

为何要学习编程

首先是工作需要。根据美国劳工统计局的数据，在 10 个增长最快、薪水最高的就业领域中，与计算相关的占了 7 个。在全球各地，程序员都很抢手，未来几年的需求量为数百万。程序员不管身处何方，只要能连上网就能赚钱。

然而，收入高、工作有保障并非学习编程的唯一原因。编程就是解决问题，而这年头需要更多能解决问题的人。你可编写让人彼此联系以及帮助他们完成工作的应用；可催生新的商业形式乃至全新的市场；可消除壁垒，向个人、行业乃至全球提供帮助，还能带来从未有过的机会。由于网络和智能手机无处不在，你编写出应用后就可与数十亿人分享。

Dropbox 创始人 Drew Houston 指出，编程是“最有可能让我们成为超人的技能”，而视频游戏开发公司 Valve 的联合创始人 Gabe Newell 说，知道如何编程让人“在他人看来就像拥有魔法”。计算机无处不在——存在于日常生活中的每台设备、每个系统、每个网络中，而代码是驱动计算机发挥作用的动力。学习编程就是学习如何在高科技时代茁壮成长。

为何要学习 Java

很多人都认为 Java 是头号编程语言，其原因有多个。首先，几乎你想象得到的每种设备都使用了它，从台式机和笔记本电脑到智能电视。同样的 Java 代码既可在 Windows 系统中运行，也可在 macOS 和 Linux 系统中运行。

其次，Java 被公司用来运行一些最庞大的企业应用。Java 是一种面向对象的编程语言，让软件工程师能够开发用于各种领域的大型应用，从制造到销售，从人力资源到财会。

再次，在全球各地的大学中，Java是最受欢迎的语言之一，学习它可迅速跟上程序员同仁的步伐。

无论你学习编程是因为兴趣爱好，还是出于副业或全职工作的需要，选择 Java 都能让你快速而轻松地上手。如果你从未学过编程语言，Java将是你的首选语言；如果你有编程经验，Java也非常值得你去学习。

本书涵盖的内容

下面概述一下本书各章的内容。

第1章将引导你安装并设置Java、Eclipse和Android Studio，你还将使用交互式shell——JShell编写第一个Java命令。

在第2章，你将编写第一个应用——猜数游戏，这是一个基于文本的命令行程序，让用户猜测一个1~100的随机数。在第3章，你将把猜数游戏升级为基于窗口的桌面应用，并添加包含标签、文本框和可单击按钮的图形用户界面（GUI）。

在第4章，你将创建第一个Android移动应用，并在其中重用前两章给猜数游戏编写的大部分代码。在第5章，你将对这个移动版猜数游戏进行多方面润色，其中包括添加设置菜单以及记录最高得分。

在第6章，你将创建应用Secret Messages，这是一个基于文本的程序，使用凯撒加密法对消息进行加密。在第 7 章，你将把这个应用升级为 GUI 的——添加一个滑条，让用户能够快速破解凯撒加密。在第8章，你将创建移动版Secret Messages应用，它提供了分享功能，让用户只需单击按钮就能通过E-mail、短信或社交媒体与朋友分享消息。

在第9章，你将着手创建本书最具视觉震撼力的应用——BubbleDraw，让用户能够在屏幕上绘制色彩缤纷的气泡。在第 10 章，你将给这个应用添加动画，让气泡四处飘散，并在到达屏幕边缘后往回弹。在第 11 章，你将添加多点触控功能，让用户能够同时在多个地方绘制气泡，并最终将其打造成一个外观专业、乐于同朋友分享的应用。

最后，附录A提供了一些小贴士，帮助你在Eclipse和Android Studio中编写Java程序时进行调试以及避免常见的错误。

需要哪些工具

在本书中，你将学习使用编程工具Eclipse和Android Studio，它们是最流行的Java开发工具，因此阅读完本书后，你马上就能编写真正的应用。最重要的是，它们都是免费的！

只要有运行Windows、macOS或Linux的台式机或笔记本电脑，并能够上网，你就可开始阅读本书。即便没有Android手机和平板电脑，你也能开发移动应用，因为Android Studio自带了免费的Android设备模拟器，可用来测试应用。当然，如果你有Android手机或平板电脑，就能直接在上面运行移动应用。

在线资源

本书所有应用的源代码都可从官网（https://www.nostarch.com/learnjava/）免费下载。如果你喜欢一对一的详尽教学视频，可前往 http://www.udemy.com/java-the-easy-way/注册在线课程，该课程将引导你循序渐进地完成每个示例和每行代码。注册该课程时，使用优惠码 BOOKHALFOFF 可打 5 折。注册该在线课程后，你可随时提问或直接给我发消息。

从现在做起

学习 Java 编程有百利而无一害，现在就开始吧！编程是打开通往全新世界大门的钥匙，而学习 Java 是迈向新职业和新未来的第一步。我的大学学生有一个共同之处，那就是他们都迈出了第一步：编写第一行代码，再编写第一个程序，再不断地学习和成长。这些你也能做到。

中国古语有云："往者不谏，来者可追。"无论你是学生还是打算换个职业，现在学习编程都正当其时。

电子书

扫描如下二维码，即可购买本书电子版。



致　　谢

如果没有 No Starch Press 编辑团队的大力支持，本书不可能付梓。这里要特别感谢 Bill Pollock、Tyler Ortman、Riley Hoffman、Jan Cash、Serena Yang、Amanda Hariri 和 Julia Borden，感谢他们不辞辛苦地编辑、审阅和营销，还有从各个方面帮助我改进本书。

感谢 Bryan Fagan 出色的技术审阅。

感谢我现在和以前的学生激励我不断创作出有趣又引人入胜的内容，尤其是 Shah Rahman、Susan Rahman、Justin Turner、Diane Turner、Jacob Elliott、Brian Murray、Aaron Walker、Robert Brown、Trent Deal、Seth Park、Simon Singh、Andrew Miller、Ctrl-Alt-Del 机器人小组、Jackson Grant、Quintin Kerns、Grace Halley、Jack Halley、Matthew Harpur 和 David Knight。还要感谢来自 150 个国家的注册我在线课程的学生，尤其是 Hayden Redd 和 Bob Watson。

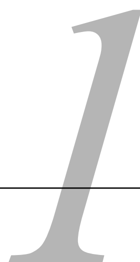
感谢给人以启迪的同事和朋友，是他们不断激励我做最好的自己：Markus Hitz、Chuck Robertson、Irene Kokkala、Tamirat Abegaz、Antonio Sanz Montemayor、Don Watkins、Eddie Mienie、Rose Procter、Ron Larson、Victor Parker、Keith Antonia、Billy Wells、Jim Goldy、James Daniel 和 Craig Gentry。

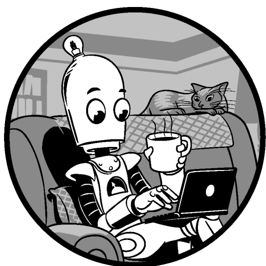
感谢岳父 Norman Petty，他对技术的热情唯有对家人的爱能出其右。感谢继父 Dale Carver 总能在百忙之中抽出时间欣赏我的 3D 打印配件。最要感谢的是我美丽的妻子 Bev 以及出色的儿子 Alex 和 Max，感谢他们在我三年内编写两本图书期间无穷的耐心。

这里要特别感谢 Kalen Cole。Kalen 在 11 岁时就抱着我编写的第一本图书 *Teach Your Kids to Code* 自学编程，还送给我颇具创意的礼物——他自己编写的代码行。Kalen，加油！每个孩子在逐渐认识到自己是谁、要成为什么样的人后，都应该以你为榜样。

目　　录

第 1 章 起　　步





本章，你将首先在计算机上安装 Java、Eclipse 和 Android Studio，再在交互式命令行 shell——JShell 中尝试执行一些命令，以熟悉基本的 Java 编程知识。

Java 是一门功能强大的多平台编程语言，可免费下载并安装到 Windows、macOS 和 Linux 系统中。作为业界标准的**集成开发环境**（integrated development environment，IDE），Eclipse 工具包可用于快速而轻松地开发 Java 应用程序。Android Studio 是一个使用 Java 开发 Android 移动应用的开发环境，让你能够开发用于手机、平板电脑等设备的移动游戏和应用。

1.1　Java 支持 Windows、macOS 和 Linux

Java 的优点之一是，你编写的 Java 程序可在任何计算机上运行，只要它安装了 Java 虚拟机（Java Virtual Machine，JVM）软件［有时也被称为 Java 运行时环境（Java Runtime Environment，JRE）］。JVM 技术让你只需编写程序一次，就可在使用任何操作系统（Windows、macOS、Linux、Android 等）的设备上运行它。

这种理念看似平淡无奇，但使用其他大多数编程语言时，你要么为 Windows、macOS、Linux 和智能手机编写不同的代码，要么针对不同的操作系统编译不同的版本。JVM 是一种运行时环境，让你无需这样做。

图 1-1 显示了同一个图形 Java 应用程序在 Windows、macOS 和 Ubuntu Linux 系统上的运行情况。

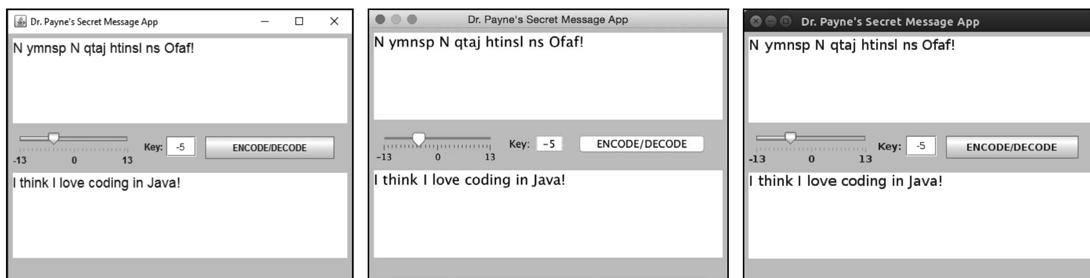


图 1-1　在三种不同的操作系统（Windows、macOS 和 Linux）中运行同样的 Java 代码

这个简单而强大的概念是 Java 被全球各地的个人和企业采用的原因之一。

1.2　安装 Java 8 for Developers 和 Java 9 for Developers

Java 开发包（Java Development Kit，JDK）是供开发人员（程序员）使用的 Java 版本，让你能够编写和编译 Java 应用程序，并将其与朋友分享、部署到企业环境中或在任何设备上运行。

我们将同时安装第 8 版和第 9 版的 JDK，这样既可获得第 8 版被广泛使用带来的好处，又可使用第 9 版提供的最新功能。

为安装 JDK 8，请执行如下步骤。

(1) 访问 http://jdk.java.net/8/。

(2) 单击单选按钮 Accept License Agreement。

(3) 在 JDK 下载列表中找到你使用的操作系统，并单击相应的链接。如果你使用的操作系统为 Windows 或 Linux，请选择 64 位版本。

(4) 双击下载到文件夹 Downloads 中的 JDK 文件以安装 JDK。

为安装 JDK 9，请访问 http://jdk.java.net/9/，再重复 JDK 8 安装过程的第 2~4 步。

注意　有关详尽的分步安装视频，请参阅与本书配套的在线课程（http://www.udemy.com/java-the-easy-way/）。

这就是需要做的所有准备工作，现在你可以从基于文本的命令行或终端编译并运行 Java 程序了。但我们还想使用 Java 来创建类似于图 1-1 所示的**图形用户界面**（graphical user interface，GUI）应用程序，为此我们将安装集成开发环境 Eclipse。

1.3　安装 Eclipse IDE for Java Developers

Eclipse 是最流行的 Java 编程 IDE 之一，它还是开源的，这意味着个人和企业都可免费使用，同时有欣欣向荣的开源社区不断地改进和支持它。还有很多其他的 Java 开发 IDE，但就开发 Java 应用程序而言，Eclipse 以易于使用著称。Eclipse 安装起来也非常容易。

为安装 Eclipse，请访问 http://www.eclipse.org/downloads/，根据你使用的操作系统下载相应的安装程序（如图 1-2 所示）再运行它。本书编写期间，最新版本为 Eclipse Oxygen。

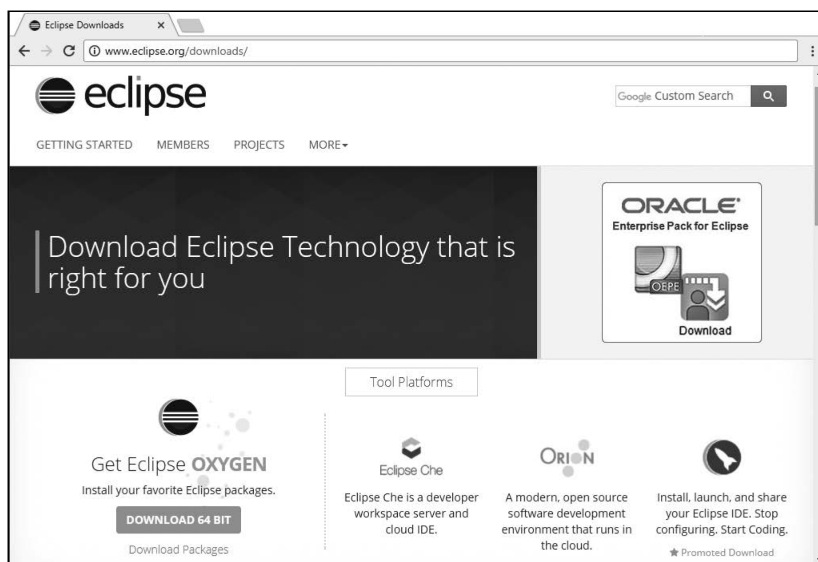


图 1-2　根据你使用的操作系统下载相应的 Eclipse IDE 安装程序

你将看到一个类似于图 1-3 所示的菜单，请选择 Eclipse IDE for Java Developers 并单击 Install。务必选择 Eclipse IDE for Java Developers（Java EE 没有本书将使用的一些功能）。

安装可能需要几分钟才能完成。

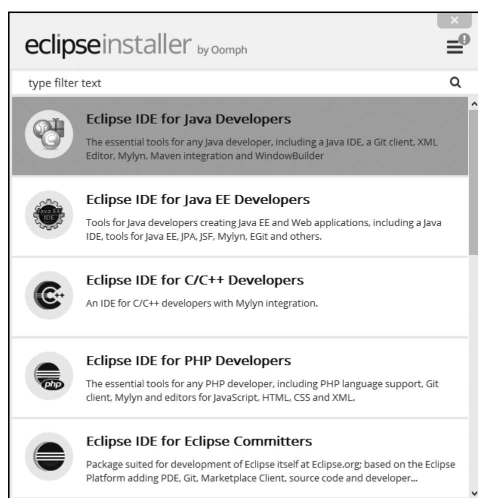


图 1-3　在 Eclipse 安装程序菜单中选择 Eclipse IDE for Java Developers

1.4　配置 Eclipse

下面来配置 Eclipse，使其更像专业开发环境：安装 WindowBuilder Editor；选择对程序员友好的颜色方案和字体。

请单击 Eclipse 图标来启动它。Eclipse 启动时，通常会让你指定**工作区**的位置——所有 Java 项目都将存储在这里，如图 1-4 所示。你可使用默认位置（在 Windows 系统中，为 C:\Users\<YourUserName>\eclipse-workspace\；在 macOS 系统中，为/Users/<YourUserName>/Documents/eclipse-workspace/；在 Linux 系统中，为/home/<YourUserName>/eclipse-workspace/），也可给 Java 工作区文件夹指定自定义位置。

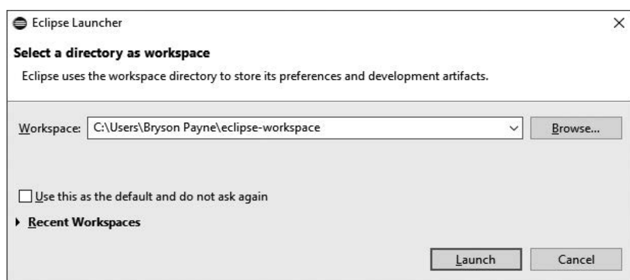


图 1-4　Eclipse 首先会询问你要将 Java 项目存储在什么地方

如果你没什么偏好，可使用默认位置 eclipse-workspace。不管你选择的是什么位置，都请记住它，因为你的所有 Java 项目都将存储在这里。如果你选中复选框“Use this as the default and do not ask again”，Eclipse Launcher 窗口就不会在你每次启动 Eclipse 时都出现。如果你要使用多个工作区，就不要选中这个复选框，这样可在启动 Eclipse 时轻松地在不同工作区之间切换。

修改 Eclipse 使其支持 Java 9

如果 Eclipse 无法启动，就还需要修改一个地方。编写本书期间，Java 9 推出没多长时间，虽然 Eclipse Oxygen 和更高的版本支持 Java 9，但有些版本需要修改配置文件 eclipse.ini 才能支持 Java 9。为完成这种修改，请执行如下步骤。

(1) 找到 Eclipse 安装文件夹。

- 在 Windows 系统中，可右击 Eclipse 快捷方式并选择“打开文件位置”。文件 eclipse.ini 与程序文件 eclipse.exe 位于同一个文件夹。
- 在 macOS 系统中，启动 Finder，并找到文件夹 Applications 中的应用程序 Eclipse，然后按住 control 键并单击应用程序图标 Eclipse，再选择“显示包内容”。依次打开“内容”和 Eclipse 文件夹，你将在文件列表中看到 eclipse.ini。
- 在 Linux 系统中，进入你的主（home）文件夹，并打开 eclipse/java-oxygen/eclipse，你将在其中找到 eclipse.ini。

(2) 右击（或按住 control 键并单击）文件 eclipse.ini，选择“打开方式”，再选择“记事本”、TextEdit 或其他文本编辑器。

(3) 在文件 eclipse.ini 末尾添加如下内容：

```
--add-modules=ALL-SYSTEM
```

(4) 将 eclipse.ini 存盘，再启动 Eclipse。从现在开始，Eclipse 应该能够正确地启动。

首次启动 Eclipse 时，你将看到一个欢迎屏幕。根据你使用的 Eclipse 版本，这个屏幕可能包含一些很有用的示例项目和教程，也可能更简单。如果你愿意，可随便单击做些探索。探索完毕后，请单击 Welcome 选项卡右上角的 X 将其关闭。

1.4.1　安装 WindowBuilder Editor

我们要对 Eclipse 做的最重要的升级是安装 WindowBuilder Editor，它让你能够创建多窗口应用程序。为此，你只需将按钮、标签和文本框等 GUI 元素拖放到应用程序的图形预览中。

有些版本的 Eclipse 安装了 WindowBuilder Editor，但我们将执行安装或更新步骤，为第 3 章开始创建 GUI 应用程序做好准备。

首先，请访问 http://www.eclipse.org/windowbuilder/并单击 Download。在下载页面中，找到与你使用的 Eclipse 版本匹配的 WindowBuilder 版本（对 Eclipse Oxygen 来说，为 4.7 版），再右击（或按住 control 键并单击）相应的链接，并复制链接地址，如图 1-5 所示。

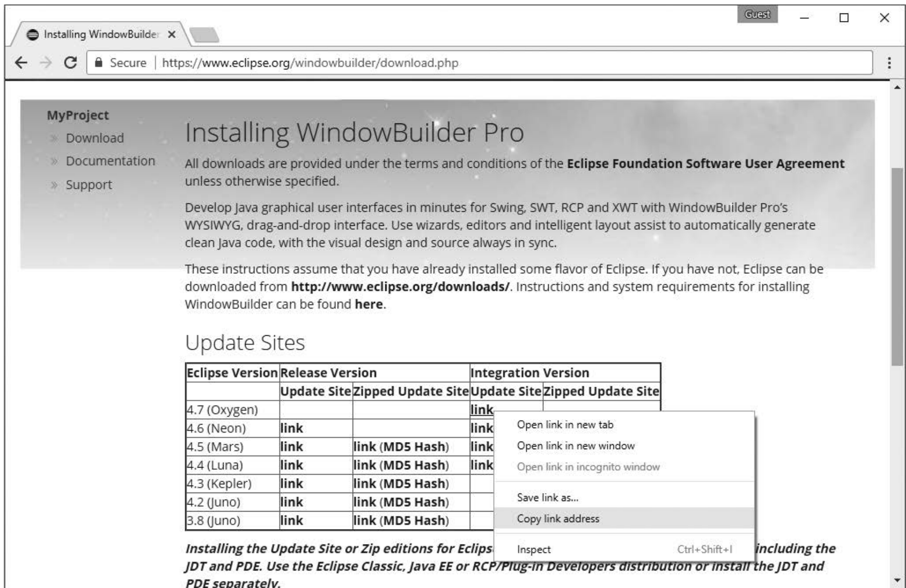


图 1-5　找到最新版 WindowBuilder Editor 的下载链接并复制其地址

回到 Eclipse 并选择菜单 Help▶Install New Software。在文本框 Work with 中，粘贴 WindowBuilder Editor 的下载 URL（对 Eclipse Oxygen 来说，为 http://download.eclipse.org/windowbuilder/WB/integration/4.7/），再单击按钮 Add...。在出现的对话框中，在文本框 Name 中输入 `WB`，如图 1-6 所示。

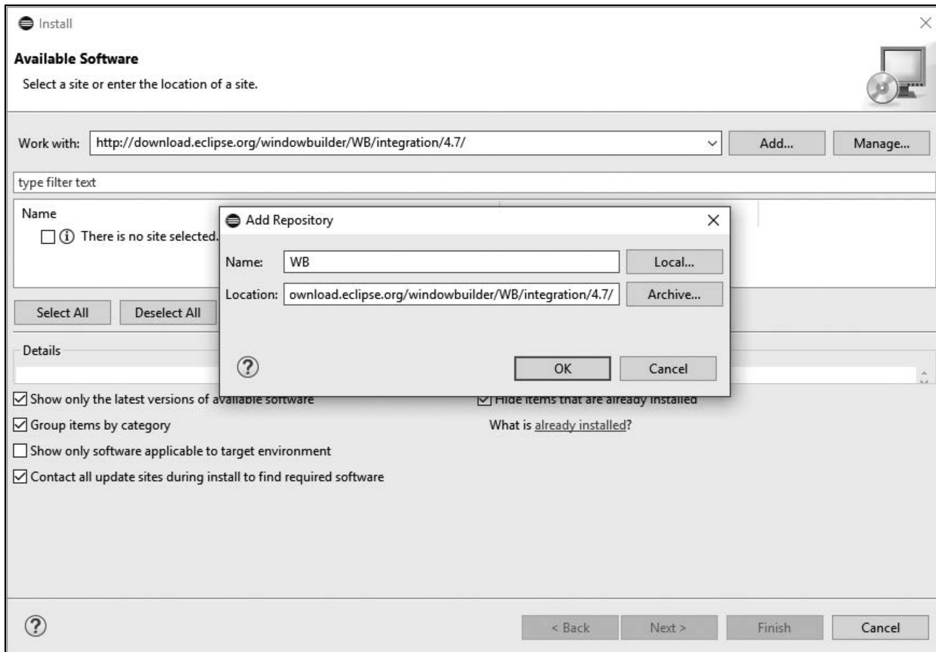


图 1-6　使用 Eclipse 菜单项 Install New Software 来添加 WindowBuilder Editor

单击按钮 OK，等安装窗口中出现复选框 WindowBuilder 后，单击按钮 Select All 安装需要的所有 WindowBuilder 组件。不断单击 Next 按钮，并在被询问时接受许可协议，再单击按钮 Finish。

这个软件可能需要几分钟才能安装完毕——Eclipse 右下角有进度指示器。安装完毕后，将询问你是否要重启 Eclipse。请单击 Restart Now 以完成 WindowBuilder 的整个安装过程。

下面来做些微调——修改背景色、文本颜色和字体，让 Eclipse 中的代码更容易阅读。

1.4.2　定制 Eclipse 的外观

安装必要的软件后，还需定制 Eclipse 的外观。为此，在 Windows 和 Linux 系统中，可选择菜单 Window▶Preferences；在 macOS 系统中，可选择 Eclipse▶Preferences。

例如，你可能想修改**主题**（调色板）和文本编辑器使用的**字号**。根据你的显示器尺寸以及编程环境，主题和字号可能对可读性、舒适程度乃至效率带来重大的影响。

在 Preferences 菜单中，展开 General 并选择 Appearance，你将看到 Theme（主题）选项。你可选择喜欢的主题，如 Classic（淡灰色背景和黑色字体）或 Dark（黑色背景和色彩丰富

的字体）。我喜欢主题 Dark，因为其字体颜色比黑色背景浅，无论是在显示器还是投影屏幕上都更容易看清楚。

在 General▶Appearance▶Colors and Fonts 中，可修改字号。请在右边的 Colors and Fonts 窗格中，选择 Basic▶Text Font，再单击 Edit 按钮，这将打开 Font（字体）对话框。请选择对你来说容易看清的字体，我喜欢 Courier New 或 Consolas。推荐你将字号设置为 18~20，并将字体样式设置为 Bold（粗体）。Font 对话框将根据你选择的字体、字号和样式显示样本，如图 1-7 所示。

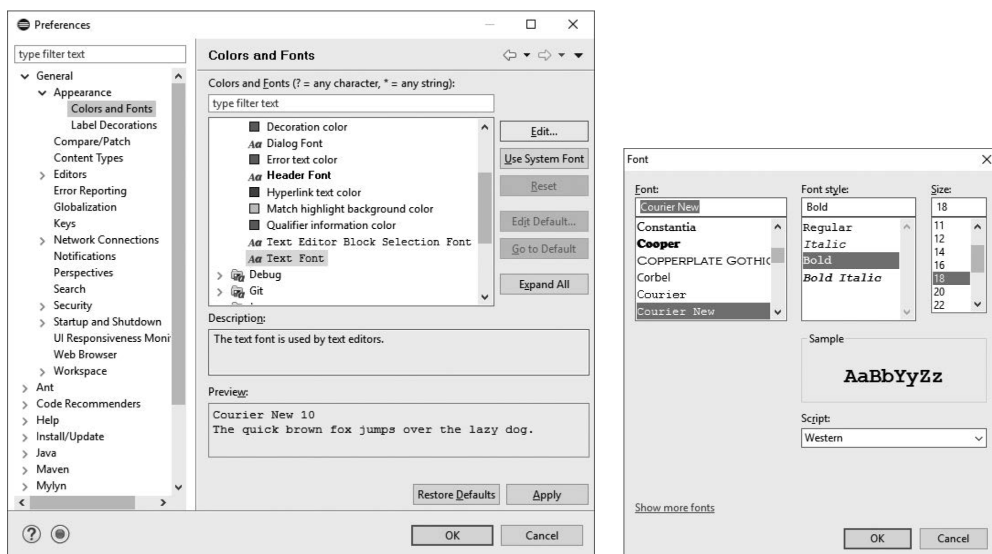


图 1-7　Colors and Fonts 首选项（左）和 Font 对话框（右）

设置好首选项后，单击按钮 OK 返回到 Eclipse 主工作区。等到第 2 章你开始在 Java 文本编辑器中编写代码时，将会发现首选项设置生效了。

1.5　安装用于开发移动应用的 Android Studio

Android Studio 是官方提供的 Android 移动应用开发环境，让你能够使用 Android 原生编程语言 Java 来设计和编写移动应用。与 Java 和 Eclipse 一样，Android Studio 也可免费下载、安装和使用。Android Studio 体量庞大，下载并安装它需要的时间从几分钟到几小时不等，这取决于你的网络连接速度。

为下载 Android Studio，请访问 http://developer.android.com/studio/并单击 Download Android Studio，如图 1-8 所示。请阅读并同意许可条款，再单击 Download Android Studio for <operating system>。

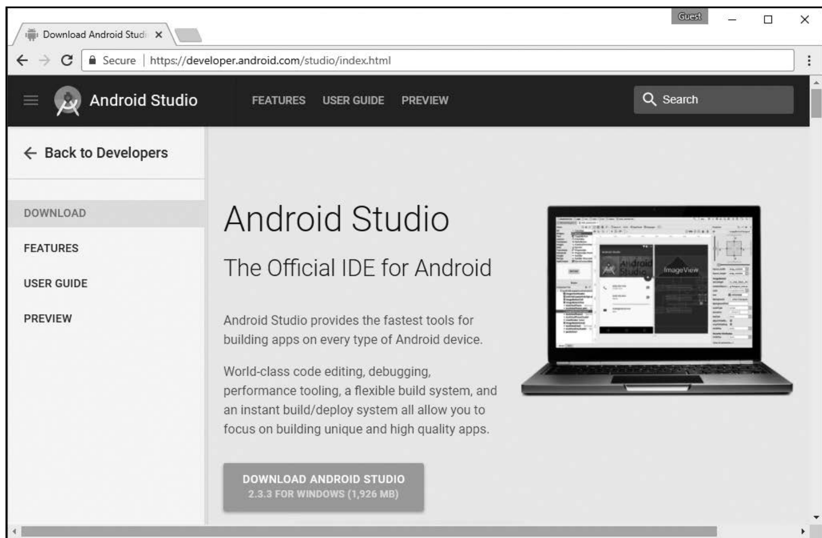


图 1-8　Android Studio 是官方的 Android 移动应用开发环境

请按说明完成安装过程。安装期间可能下载其他的**软件开发包**（software development kit，SDK）组件，而下载完所有组件后，可能还需要几分钟才能安装完毕。

1.6　使用 JShell 熟悉 Java

搭建好编程环境后，我们使用 Java **交互式解释器** JShell 来检查安装情况。JShell 非常适合用来了解 Java 的工作原理，因为它在你输入代码后立即提供反馈。在 Java 9 之前，Java 程序员必须输入完整的程序，再编译并运行，才能看到结果。现在有了 JShell，我们输入单行 Java 代码（如 `System.out.println("Hello,Java!")`并按回车键后，就能在屏幕上看到输出，如图 1-9 所示。

```
|  Welcome to JShell -- Version 9-ea
|  For an introduction type: /help intro

jshell> System.out.println("Hello, Java!")
Hello, Java!

jshell>
```

图 1-9　Java 9 中的 JShell 让你能够在交互式命令行 shell 中快速尝试代码

在 JShell 中可执行任何合法的 Java 语句，这让它非常适合用来学习基本的 Java 编程知识。下面就来看看。

1.6.1　运行 JShell

你可从命令行运行 JShell，也可创建快捷方式并通过它来运行 JShell。这里将同时介绍这两种方法，以防有一种方法对你来说不管用。

首先，你必须安装 JDK 9。要确定是否安装了 JDK 9，可在命令行运行一个命令。

要在 Windows、macOS 或 Linux 系统中启动命令行界面，可像下面这样做。

- 在 Windows 系统中，这样打开命令提示符：单击“开始”按钮，在搜索框中输入 cmd 并按回车键或单击命令提示符图标。
- 在 macOS 系统中，打开 Launchpad 并在搜索框中输入 terminal，再单击应用程序图标 Terminal。
- 在 Linux 系统中，搜索 terminal，再单击应用程序图标 Terminal。

此时将出现一个命令提示窗口。在提示符下输入 java -version，Java 将指出安装的是哪个 JDK 版本。在 Windows 系统中，你将看到类似于下面的输出：

```
C:\Users\Payne> java -version
java version "9-ea"
Java(TM) SE Runtime Environment (build 9-ea+153)
Java HotSpot(TM) 64-Bit Server VM (build 9-ea+153, mixed mode)
```

在 macOS 或 Linux 系统中，你将看到类似于下面的输出：

```
Payne:~ payne$ java -version
java version "1.9.0_33"
Java(TM) SE Runtime Environment (build 1.9.0_33)...
```

只要输出像上面这样指出安装的 Java 版本为 9 或 1.9，就可运行 JShell 了。如果你要从命令行运行它，可继续阅读下面的“从命令行运行 JShell”。如果输出中显示的是更早的 Java 版本，如 Java 1.8，请按 1.2 节的介绍安装 JDK 9。如果安装 JDK 9 后，前述命令的输出依然未包含版本 9，或者你更喜欢使用快捷方式运行 JShell，请跳到“从快捷方式运行 JShell”。

1. 从命令行运行 JShell

要从命令提示符运行 JShell，可在其中输入 jshell 并按回车。Java 将显示 JShell 欢迎消息和提示符，它们在 Windows 系统中类似于下面这样：

```
C:\Users\Payne> jshell
|  Welcome to JShell -- Version 9-ea
|  For an introduction type: /help intro

jshell>
```

在 macOS 或 Linux 系统中，消息和提示符类似于下面这样：

```
Payne:~ payne$ jshell
|  Welcome to JShell -- Version 9-ea
|  For an introduction type: /help intro

jshell>
```

如果出现了提示符 jshell>，你就可直接进入 1.6.2 节。如果安装 JDK 9 后命令 jshell 依然不管用，请尝试执行下一小节介绍的步骤。

2. 从快捷方式运行 JShell

如果你无法从命令行启动 JShell，或者你想从桌面快捷方式启动它，请按下面的步骤进入 JDK 9 安装目录下的文件夹 bin，再找到 JShell 并创建快捷方式。文件夹名称 bin 是 binaries（二进制文件）的简写；所谓**二进制文件**指的是使用计算机语言（只包含 0 和 1）编写的程序。

注意　遗憾的是，在 Linux 系统中，这种方法不一定管用，但命令行方法应该管用。

JShell 位于下面的目录中。

- Windows 系统：C:\Program Files\Java\jdk-9\bin\jshell.exe。
- macOS 系统：/Library/Java/JavaVirtualMachines/jdk-9.jdk/Contents/ Home/bin/jshell。

在你的计算机上，JDK 9 安装目录可能是 jdk-1.9.x，而不是 jdk-9。请进入 JDK 9 安装目录下的文件夹 bin，并找到文件 JShell，如图 1-10 所示。

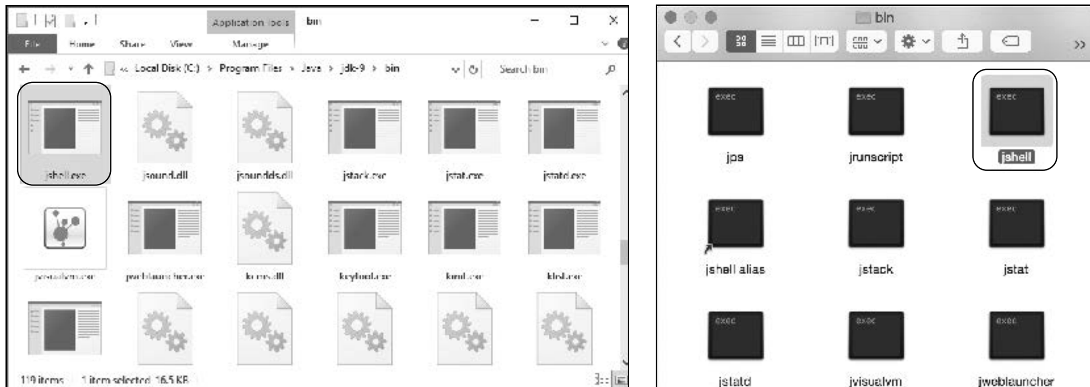


图 1-10　在 Windows 系统（左）和 macOS 系统中，JDK 9 安装目录下文件夹 bin 中的程序文件 JShell

你可双击 JShell 文件的图标来启动 JShell，也可创建一个快捷方式，这样就可直接从桌面启动 JShell 了。

- 在 Windows 系统中，右击 jshell.exe 并选择“发送到”▸“桌面快捷方式”。
- 在 macOS 系统中，按住 control 键并单击文件 jshell，再选择“创建别名”。将出现一个名为 jshell alias 的文件，请将其拖放到桌面。

现在，无论什么时候想启动 JShell，都只需双击桌面上的图标，然后就可以编写 Java 代码了。

1.6.2　在 JShell 中使用 Java 表达式

表达式是值（如数字或文本）和运算符的任何组合，条件是其结果为另一个值。**运算符**执行加减乘除等运算；在 Java 中，用来表示加减乘除运算的符号分别为+、-、*和/。我们来尝试一个

简单的数学表达式。在 JShell 提示符下输入 2+2 并按回车：

```
jshell> 2+2
$1 ==> 4
```

JShell 指出，表达式 2+2 的值为 4。$1 是一个**临时变量**，JShell 创建这样的变量来存储值；在这里，变量$1 用于临时存储 4，以便你以后使用。要获悉变量的值，可在 JShell 中输入它。例如，如果你现在输入$1，JShell 将指出$1 存储的值为 4：

```
jshell> $1
$1 ==> 4
```

再来尝试一个表达式。这次我们合并两个文本字符串。**字符串**是用引号括起的字符，用于显示单词、名称和其他文本。要合并字符串，可使用运算符+：

```
jshell> "Your" + "Name"
$3 ==> "YourName"
```

如你所见，又创建了一个临时变量，这里是$3。$后面的数字指出了相应的表达式是在第几行输入的。这是我输入的第三个代码片段，因此 JShell 将"YourName"存储在变量$3 中。

另外请注意，Java **拼接**两个字符串（或者说将它们相加）时，不会在它们之间添加空格。如果你要在拼接的两个字符串之间添加空格，必须将其包含在双引号内，如下所示：

```
jshell> "Your" + " " + "Name"
$4 ==> "Your Name"
```

只要是合法的 Java 表达式，就可在 JShell 中计算它。我们再来尝试几个表达式。要编辑以前输入的语句，可按向上箭头键，JShell 将显示你输入的最后一条语句，让你能够对其进行编辑。编辑完毕后，可按回车键再次运行它。不断按向上箭头键，可遍历以前输入的所有语句，直到到达你输入的第一行。

1.6.3　在 JShell 中声明 Java 变量

计算由简单值组成的表达式很好，但你通常想将值存储在变量中，以便以后能够使用。JShell 会不断地替我们自动创建变量，如前一节的$1 和$3，但你也可自己创建（**声明**）变量。

1. 数字变量

我们来创建一个名为 x 的整数变量，并将值 42 存储到其中。Java 使用类型为 int 的变量来存储整数值，因此请在 JShell 提示符下输入 int x = 42：

```
jshell> int x = 42
x ==> 42
```

在 Java 中，等号（=）为**赋值运算符**，即用于给变量**赋值**。JShell 做出响应，让你知道变量 x

包含的值为 42。现在，如果你要使用这个值，只需引用变量 x 即可。我们来看看 x * 2（x 乘以 2）等于多少：

```
jshell> x * 2
$6 ==> 84
```

Java 将星号（*）用作乘法运算符，而 JShell 告诉你 x * 2 等于 84。但获取 x 的值并将其与 2 相乘时，修改了 x 的值吗？为获悉这一点，我们输入 x：

```
jshell> x
x ==> 42
```

变量 x 的值没变，还是 42。这意味着使用变量存储的值时，不会修改它的值。

那么如何修改变量存储的值呢？只需再次使用赋值运算符。请在 JShell 提示符下输入 x = x + 7：

```
jshell> x = x + 7
x ==> 49
```

我们获取 x 的值，将其与 7 相加，并将结果存储到变量 x 中。从现在开始，每当你询问 x 的值时，都将得到 49，直到你再次修改它。你想什么时候修改变量的值就可什么时候修改，这正是它们被称为变量的原因所在。

下面来尝试几个不同类型的变量。前面看过了整数值，下面来尝试一个小数（**浮点数**）。Java 使用类型为 double（意思是**双精度浮点数**）的变量来存储小数，因此我们创建一个名为 meters 的 double 变量，并在其中存储小数值 1.83：

```
jshell> double meters = 1.83
meters ==> 1.83
```

对 Java 来说，小数处理起来与整数一样容易。我们来做点数学运算，将米转换为厘米：

```
jshell> double centimeters = meters * 100
centimeters ==> 183.0
```

我们将 meters 的值乘以 100，并将结果存储到变量 centimeters 中。

Java 还支持其他几种数字类型，但 int 和 double 是最常用的。每当你遇到没有见过的类型时，都可启动 JShell 并尝试在其中使用它们。

2. 字符串变量

类型为 String 的变量用于存储文本字符串。我们来定义一个名为 myName 的 String 变量，并在其中存储一个姓名，如下所示（你可存储你的姓名）：

```
jshell> String myName = "Bryson Payne"
myName ==> "Bryson Payne"
```

与前面给数字变量赋值一样，我们也使用等号来赋值。

注意 在 Java 中，变量、方法和类的名称是区分大小写的。全小写的 myname 不同于全大写的 MYNAME，而这两个都不同于 myName。在 Java 中，约定（通行的做法）是使用**骆驼拼写法**，即名称中每个单词的首字母都大写（第一个单词除外），如 myName 或 thisIsASillyNameButShowsCamelCase，这让每个首字母看起来都像驼峰。对于类名，除采用骆驼拼写法外，还将第一个字母也大写。

下面来使用存储在变量 myName 中的值。假设你获得了证书或学位，需要在你的姓名后面添加一些字符，如下所示：

```
jshell> myName + ", PhD"
$12 ==> "Bryson Payne, PhD"
```

请注意，这里没有使用赋值运算符，因此存储在 myName 中的值应该还是你的姓名，而不包含新增的字符。

下面来修改存储在变量 myName 中的值，在你的姓名前加上正式的头衔，就像在信封或邀请函中那样：

```
jshell> myName = "Dr. " + myName
myName ==> "Dr. Bryson Payne"
```

JShell 显示了存储在变量 myName 中的新值。下一节将继续使用数字变量和字符串变量，并介绍如何在 Java 程序中将值输出到屏幕上。

1.6.4 在 Java 中打印输出

到目前为止，我们都这样计算表达式：在 JShell 中输入它们以查看结果。但在实际编程中，情况通常不是这样的。在 Java 程序中，当你逐行输入代码时，在屏幕上看不到任何反馈。

要将内容打印到屏幕上，可使用打印函数，如 System.out.println()，它将一行输入打印到系统控制台（即屏幕）。如果你没有关闭 JShell，可像下面这样打印 x 的值：

```
jshell> System.out.println(x)
49
```

如果这样做时出错，请使用语句 int x = 49 重新声明变量 x，再执行刚才的打印语句。

注意，现在 JShell 显示的不是 x ==> 49，这是因为你并没有让它计算表达式。println()语句让 JShell 打印括号内的值（这里为变量 x），因此 JShell 只显示 49。

下面来尝试打印字符串，为此请输入下面的语句：

```
jshell> System.out.println("Hello, " + myName)
Hello, Dr. Bryson Payne
```

只要现在依然可使用前一节声明的变量 myName，Java 就会向你发出问候。

每当需要将信息打印到屏幕上供用户查看时，都可使用 System.out.println()语句，它打印你指定的内容。

1.6.5 JShell 命令

JShell 使用起来很容易，你可能都不愿离开，但你终究需要去做其他的工作，如使用 Java 编写激动人心的桌面和移动应用。下面来看看 JShell 提供的命令，包括退出 JShell 的命令。

在 JShell 提示符下输入/help 并按回车：

```
jshell> /help
```

JShell 将列出它支持的所有特殊命令。每个命令都以斜杠（/）打头，这指出我们要与 JShell 程序（而不是 Java）交流。下面列出了 JShell 支持的部分命令：

```
|    /list [<name or id>|-all|-start]    -- list the source you have typed
|    /edit <name or id>                  -- edit a source entry referenced by name or id
|    /save [-all|-history|-start] <file> -- save snippet source to a file
|    /open <file>                        -- open a file as source input
|    /vars [<name or id>|-all|-start]    -- list the declared variables and their values
|    /imports                            -- list the imported items
|    /exit                               -- exit jshell
|    /reset                              -- reset jshell
|    /history                            -- history of what you have typed
|    /help [<command>|<subject>]         -- get information about jshell
```

请尝试其中一些命令，如/list（它列出你已输入的源代码）。注意，必要时 JShell 会在行尾加上分号——Java 使用分号来分隔程序中的语句。命令/history 显示你输入的所有内容，其中包括命令，如/help、/list 乃至/history。

你在第 2 章编写 Java 程序时，将在文件中进行，这要求你反复存盘并在需要时重新打开文件。然而，在 JShell 中，一旦你关闭 JShell 窗口，输入的所有内容都将消失——除非你将其保存。所幸 JShell 提供了保存、打开和编辑代码片段的功能。要将你在 JShell 中创建的代码存盘，可使用命令/save 并指定文件存储位置：

```
jshell> /save ~/Desktop/filename.txt
```

波浪字符（~）表示你的用户目录，因此上述命令将把你从打开 JShell（或最后一次执行命令/reset）开始输入的所有代码，存储到桌面的一个文件中。如果你查看桌面，将看到这个新文件。

要打开文件，可使用/open 命令并告诉 JShell 到哪里去查找文件：

```
jshell> /open ~/Desktop/filename.txt
```

JShell 将打开这个文件并运行其中的代码。

每当你要将编写的代码片段保存供以后使用时，都可使用/save 将其保存，再在以后启动JShell 后使用/open 来加载它们。

要开始新的代码片段，可使用命令/reset。JShell 将只保存你在执行命令/reset 后输入的代码，但你随时都可打开已保存的文件。请尝试下面的代码，这个简单的示例演示了如何保存文件、重置和打开文件：

```
❶ jshell> /reset
   |  Resetting state.
❷ jshell> System.out.println("Hello, Java!")
   Hello, Java!
   jshell> System.out.println("My name is Bryson, nice to meet you!")
   My name is Bryson, nice to meet you!
❸ jshell> /save ~/Desktop/myJava.txt
❹ jshell> /reset
   |  Resetting state.
❺ jshell > /open ~/Desktop/myJava.txt
❻ Hello, Java!
   My name is Bryson, nice to meet you!
```

注意 带数字的圆圈标出了重要的代码行，但它们并不是代码的组成部分。

首先，你需要重置 JShell❶，以免保存之前输入的代码。重置后，就可编写程序了；这里的程序包含两条打印语句❷。接下来，你将这个程序保存到桌面❸。即便重置 JShell❹后，依然可以使用命令/open❺加载在命令/save 前输入的两行代码，并运行它们❻。你可使用/edit 来修改这些代码，并在修改完毕后再次使用/save 将其保存。

无论代码片段包含多少行，JShell 都能够存储并重新加载它。

探索完 JShell 后，可使用命令/exit 将其关闭：

```
jshell> /exit
Goodbye
```

JShell 很有礼貌，离开时会跟你说再见。

1.7 小结

在本章中，你不仅安装了 Java、Eclipse 和 Android Studio，还开始学习 Java 了——在 JShell 中尝试各种命令。对无处不在的 Java 来说，JShell 是一项激动人心的改进。无论是对初学者还是久经沙场的开发人员来说，JShell 都是一个绝妙的工具。

对编程新手来说，JShell 消除了学习基本 Java 命令的壁垒，可方便他们进行探索。对经验丰富的程序员来说，JShell 让他们能够快速测试代码片段，并在屏幕上立即看到结果。作为教员和程序员，JShell 的面世让我激动万分，这对 Java 的未来意义重大，同时数百万的程序员也将受益

于这组重要的新工具。

你为开发 Java 桌面和移动应用搭建好了环境，学习了如何在 JShell 中测试代码。你还对后面编写 Java 桌面和移动应用时将用到的众多编程概念有了一定的认识。下面概述了你在本章完成的工作：

- 安装 Java JDK 8 和 Java JDK 9；
- 安装 Eclipse IDE for Java Developers 和 WindowBuilder Editor；
- 安装用于开发移动应用的 Android Studio；
- 从命令行以及从文件夹 JDK9/bin 运行 JShell；
- 在 JShell 中计算包括数字和字符串的 Java 表达式；
- 在 Java 中声明用于存储整数、小数和字符串的变量；
- 使用 `System.out.println()`将输出打印到屏幕；
- 使用`/reset`、`/edit`、`/save`、`/open` 和`/exit` 等 JShell 命令；
- 在 JShell 中保存和打开文件。

在下一章，我们将创建第一个完整的 Java 应用程序——猜数游戏。在随后的几章中，我们将创建基于文本的程序、桌面应用程序及其 Android 移动版。

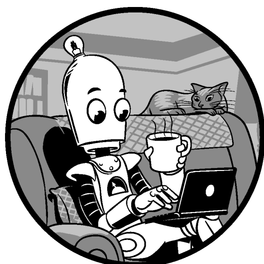
在这个过程中，将使用你在本章通过 JShell 学到的编程概念，这包括表达式、变量、输出等。无论你是第一次还是第 800 次编写程序，也无论你是为完成工作而编写桌面应用程序还是出于好玩而编写移动游戏，前述元素都不可或缺。

万事俱备，我们进入第 2 章撸起袖子干吧！

第 2 章

创建猜数游戏

2



下面来使用 Java 编写一个有趣又好玩的游戏——猜数游戏。我们将把这个游戏编写成**命令行应用程序**，即基于文本的程序，如图 2-1 所示。这个程序让用户猜 1~100 的数字，用户每次做出猜测后，程序都将指出他猜大了、猜小了还是猜对了。

```
Command Prompt - java HiLo
Guess a number between 1 and 100:
50
50 is too high. Try again.
Guess a number between 1 and 100:
25
25 is too low. Try again.
Guess a number between 1 and 100:
37
37 is too low. Try again.
Guess a number between 1 and 100:
43
43 is correct! You win!
It only took you 4 tries! Good work!
Would you like to play again (y/n)?
y
```

图 2-1　基于文本的猜数游戏

知道这个游戏的工作原理后，你只需完成每个步骤的编码工作即可。我们将首先制订粗略的计划，再编写这个游戏的极简版本。对目标心中有数并知道游戏的玩法后，你就能带着目的轻松地学习编码技能，还能在编写好游戏后立即把玩一番。

2.1　游戏步骤规划

我们来考虑一下，要让猜数游戏正确运行，需要完成哪些编码步骤。这款游戏的基本版本需要做如下事情。

(1) 生成一个 1~100 的随机数，让用户去猜。

(2) 显示一条**提示语**（即一行文本），让用户猜测这个数是多少。

(3) 接受用户的猜测。

(4) 将用户猜测的数字与生成的随机数进行比较，看用户猜测的数字大了、小了还是正确。

(5) 将结果显示到屏幕上。

(6) 不断让用户去猜测，直到猜对为止。

(7) 询问用户是否要接着玩。

我们将从这个基本结构着手。在本章后面的编程练习 2 中，你将再添加一项功能——告诉用户猜了多少次才猜对。

2.2　新建 Java 项目

在 Eclipse 中开发新 Java 应用程序时，首先要做的是创建一个项目。为此，可选择菜单 File▶New▶Java Project（也可选择菜单 File▶New▶Project，再在 New Project 向导中选择 Java▶Java Project），这将打开 New Java Project 对话框，如图 2-2 所示。

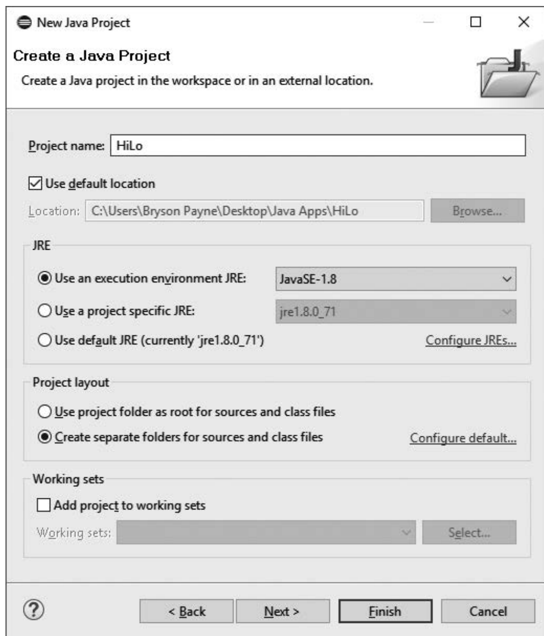


图 2-2　猜数游戏的 New Java Project 对话框

在文本框 Project name 中输入 `HiLo`。请注意，在 Java 中大小写很重要，我们将养成将项目名、文件名和类名的首字母大写的习惯，因为这是 Java 惯例。另外，我们还将使用骆驼拼写法，这里为 HiLo，因为 Hi 和 Lo 是两个单词。保留其他设置不变，并单击 Finish 按钮。根据你使用的 Eclipse 版本，可能询问你是否要打开 Java 透视图。在 Eclipse 中，**透视图**是为特定语言配置的工作区。如果被这样询问，请单击 Yes，告诉 Eclipse 你希望将工作区配置成方便使用 Java 进行编程的样子。

2.3 创建 HiLo 类

Java 是一种**面向对象的编程语言**。面向对象的编程语言使用**类**来设计可重用的代码片段。类犹如模板，让创建**对象**（即类实例）的工作更轻松。如果说类是饼干模子，那么对象就是饼干。与饼干模子一样，类也是可重用的，因此创建类后，就可反复使用它来创建对象——想创建多少就创建多少。

这里的猜数游戏只有一个类文件，它创建一个猜数游戏对象，其中包含游戏所需的全部代码。我们将把这个类命名为 `HiLo`。大小写很重要，而类名 `HiLo` 遵守了多个 Java 命名约定。通常，类名的第一个字母都大写，`HiLo` 遵守了这种约定。另外，在组成类名的单词之间，不能有空格、连字符或特殊字符。最后，对于包含多个单词的类名，我们使用骆驼拼写法：每个单词的首字母都大写，如 `HiLo`、`GuessingGame` 和 `BubbleDrawApp`。

为创建新类 `HiLo`，首先在 Eclipse 工作区左边的 Package Explorer 窗格中找到文件夹 HiLo。单击左边的箭头将这个文件夹展开，你将看到一个名为 src（表示 source code，即源代码）的子文件夹。包含 Java 程序的所有文本文件都将放在这个文件夹中。

右击文件夹 src 并选择 New▶Class，如图 2-3 所示。

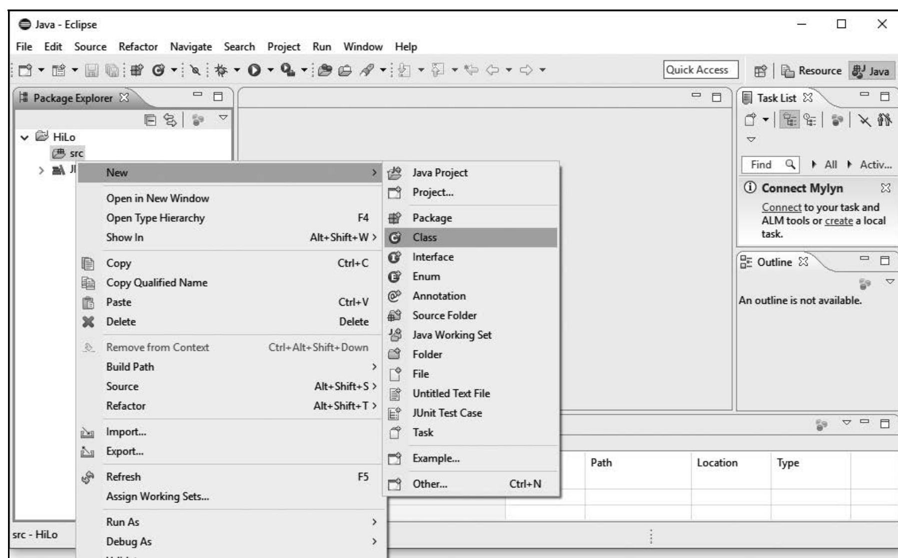


图 2-3　在猜数游戏中新建一个类文件

这将打开 New Java Class 对话框，如图 2-4 所示。在文本框 Name 中输入 `HiLo`。在“Which method stubs would you like to create?”部分，选中复选框“public static void main(String[] args)”，这告诉 Eclipse 我们要编写方法 `main()`，这样 Eclipse 将包含方法 `main()` 的**存根**（stub，或骨架），我们可在其中填充所需的代码。**方法**是位于对象或类中的函数；要将应用程序作为独立的程序运行，方法 `main()` 必不可少。

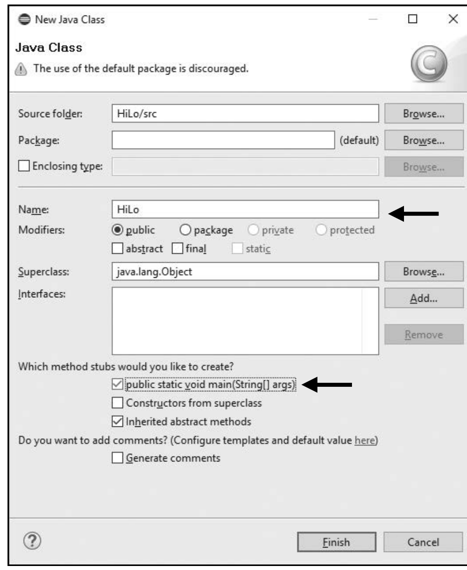


图 2-4　将新建的 Java 类命名为 `HiLo` 并指定要创建方法 `main()`

在 New Java Class 对话框中单击 Finish 按钮，你将看到一个名为 HiLo.java 的新文件，其中包含如代码清单 2-1 所示的代码。这个 Java 文件是猜数游戏的轮廓，我们将通过编辑它来编写这个游戏。

代码清单 2-1　Eclipse 为猜数游戏类 `HiLo` 生成的代码

```
❶ public class HiLo {
    ❷ public static void main(String[] args) {
          // TODO Auto-generated method stub

      }
  }
```

这些代码都是 Eclipse 创建的。`HiLo` 类是公有的❶，这意味着可从命令行或终端运行它。

Java 使用大括号（{和}）将语句编组。左大括号（{）指出了构成 `Hilo` 类体的语句块的起始位置，而右大括号（}）指出了语句块的结束位置。在这个类中，包含方法 `main()`❷，它是执行这个类时将运行的方法。

在定义方法 `main()`的大括号内，是一个注释行，它以两个斜杠（`//`）打头。注释是供人类阅读的，会被计算机忽略，因此可使用注释来指出代码的作用或添加说明。你可将代码清单 2-1 中的 `TODO` 注释删除。

2.3.1　生成随机数

这个游戏的第一项编程任务是生成一个随机数。为此，我们将使用 `Math` 类，它包含一个方

法，可用于生成 0.0~1.0 的随机浮点数（小数）。我们将把这个小数转换为 1~100 的**整数**。Math 是一个内置类，包含很多很有用的数学函数，如高级科学计算器中的函数。

在方法 main()中，添加如代码清单 2-2 所示的注释和代码行（新增的代码为黑色，既有的代码为灰色）。

代码清单 2-2　生成 1~100 的随机数的代码

```
public class HiLo {
    public static void main(String[] args) {
        // 生成要让用户去猜的随机数
        int theNumber = (int)(Math.random() * 100 + 1);
    }
}
```

首先，需要创建一个变量，用于存储要让用户去猜的随机数。由于这个游戏将让用户猜测 1~100 的整数，我们将这个变量的类型声明为 int（整数），并将其命名为 theNumber。等号（=）将一个值赋给新变量 theNumber。我们使用内置函数 Math.random()生成一个 0.0~1.0（不含）的随机数。为将取值范围扩大到 0.0~100.0（不含），我们将这个随机数乘以 100。然后，我们再加 1，将取值范围改为 1.0~101.0（不含）。

上述语句中的(int)被称为**强制类型转换**，将数字从小数类型转换为整数。在这里，将删除小数部分，得到一个 1~100 的整数。接下来，将整个数字（用户要猜的数字）存储到变量 theNumber 中。最后，我们添加一个分号（;），指出语句到此结束。

现在，可添加一条 System.out.println()语句，将生成的数字打印出来：

```
        int theNumber = (int)(Math.random() * 100 + 1);
        System.out.println( theNumber );
    }
}
```

添加这行代码后，可运行这个程序，它将生成并打印一个数字。请单击菜单栏下方的绿色 Run 按钮，以编译并运行这个程序，如图 2-5 所示。你也可以选择菜单 Run ▶ Run。

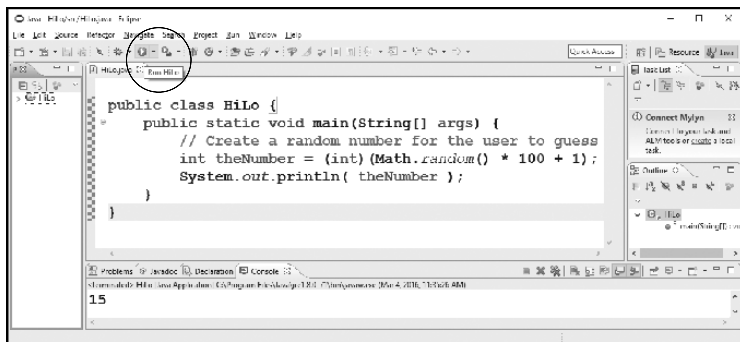


图 2-5　在屏幕上打印一个随机数

随机数将出现在屏幕底部的一个小型控制台窗口中，如图 2-5 所示。如果你再次运行这个程序，将看到另一个位于 1~100 的数字。

现在正是把玩这个程序的大好时机。请尝试让它生成 1~10、1~1000 乃至 1~1 000 000 的随机数。Java 能够接受大到 1 000 000 000 的数字。只需记得，写数字时不要加逗号。在 Java 中，1,000 将变成 `1000`，1,000,000 将变成 `1000000`。然而，刚开始玩这个游戏时，你可能不想去猜 1~1 000 000 的数字，因此把玩完后别忘了将这行代码恢复原样。

注意　别忘了经常将代码存盘。每当你运行程序时，Eclipse 都会自动将其存盘，但最好每编写几行代码后就存盘。事实上，每编写一行代码后都按 CTRL-S（或⌘-S）存盘是个很不错的习惯。我从未听过哪位程序员说他后悔频繁地存盘了，但我多次遇到过代码因为未存盘而丢失的情况，这可不好玩。请频繁地存盘吧，另外别忘了，如果你输入了错误的代码，或者不小心删除了一段代码，可选择菜单 Edit▶Undo。

2.3.2　获取来自键盘的用户输入

现在来添加让用户猜数的代码，为此需要**导入**其他一些 Java 功能。Java 自带了很多库和包，你可在自己的项目中使用它们。库和包是其他人创建的代码，通过导入它们，你就可以使用新功能，从而更轻松地创建程序。每当需要包和库时，都可使用 `import` 语句来导入它们。

就这个猜数游戏而言，我们需要能够接受来自键盘的用户输入。`java.util utilities` 包中的 `Scanner` 类提供了多个可帮助处理键盘输入的函数。下面在程序中导入 `Scanner` 类，为此在文件 HiLo.java 开头（即代码行 `public class HiLo` 前面）添加如下语句：

```
import java.util.Scanner;

public class HiLo {
```

这行代码从主要的 Java 工具包中导入 `Scanner` 类及其所有功能。`Scanner` 包含函数 `nextLine()` 和 `nextInt()`，其中前者接受一行键盘输入，而后者将文本输入转换为可用于比较和计算的整数。要使用 `Scanner` 来获取键盘输入，必须让它将键盘作为输入源。

需要在程序中做其他事情前这样做，因此在方法 `main()`开头添加如下代码行：

```
public class HiLo {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        // 生成一个要让用户去猜的随机数
        int theNumber = (int)(Math.random() * 100 + 1);
```

这行代码创建一个名为 `scan` 的 `Scanner` 对象，它从计算机键盘（`System.in`）获取输入。

虽然这行代码创建了对象 `scan`，但没有让用户输入。要让用户进行猜测，需要提示用户输入

一个数字。然后，我们将获取用户通过键盘输入的数字，并将其存储在一个变量中，以便将其同计算机生成的随机数 theNumber 进行比较。对于用于存储用户猜测的变量，我们给它指定一个好记的名称，如 guess。请添加如下代码行：

```
public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);
    // 生成一个要让用户去猜的随机数
    int theNumber = (int)(Math.random() * 100 + 1);
    System.out.println( theNumber );
    int guess = 0;
```


这条语句**声明**了一个类型为 int（在 Java 中表示整数）、名称为 guess 的变量，并将其初始值设置为 0。在有些编程语言中，必须在不同的代码行中声明和初始化变量，但 Java 允许程序员在同一行中声明并初始化变量。Java 要求声明变量时必须指定其**类型**，或可存储什么样的信息。用户的猜测为整数，因此我们将 guess 声明为 int 类型。

接下来，需要提示用户输入猜测。可向控制台窗口（命令行窗口）打印一行文本，让用户知道程序可接受输入了。基于文本的屏幕是计算机系统的一部分，可通过 System 类来访问它，就像获取键盘输入时一样。但我们要向用户显示信息，而让我们能够访问命令行控制台以显示输出的对象是 System.out。对象 System.in 让我们能够从键盘获取文本输入，而 System.out 让我们能够将文本输出到屏幕。用于打印文本行的函数为 println()：

```
    int guess = 0;
    System.out.println("Guess a number between 1 and 100:");
```

这里使用了**句点表示法**，即先列出类或对象，再加上句点以及该类或对象的方法或属性。调用方法时，必须使用句点表示法，让 Java 知道它属于哪个对象或类，如 Math.random()。**属性**是存储在对象或类中的值。

例如，System 是表示计算机系统的类，而 System.out 是包含在 System 类中的命令行屏幕对象，因为显示器是整个计算机系统的一部分。System.out.println()是一个方法，用于打印一行文本。随着你往下学习，将更多地使用句点表示法。

让用户知道计算机要求他提供什么样的输入后，该从键盘获取用户的猜测了。为此，我们将使用前面创建的 Scanner 对象 scan。Scanner 类有一个 nextInt()方法，它获取用户通过键盘输入的下一个 int 值。我们将用户的猜测存储到前面创建的变量 guess 中：

```
    System.out.println("Guess a number between 1 and 100:");
    guess = scan.nextInt();
```

这条语句等待用户通过键盘输入内容（但愿是 1~100 的整数；第 3 章将介绍如何确保用户输入的是有效的数字）并按回车键。方法 nextInt()获取用户输入的文本字符串（如"50"），将其转换为相应的数字值（50），再将结果存储到变量 guess 中。请将至此所做的修改存盘。

2.3.3　让程序打印输出

为检查程序能否正确运行，我们还可再添加一条 println()语句：

```
        guess = scan.nextInt();
        System.out.println("You entered " + guess + ".");
```

这行代码也使用了方法 System.out.println()，但同时输出文本和数字。如果用户猜测的数字为 50，我们希望输出为"You entered 50."。为此，我们编写了一条 println()语句，它将文本和存储在变量 guess 中的数字合并起来。

在 Java 中，可使用运算符+来拼接文本字符串。我们首先使用双引号来指定要输出的文本（"You entered "）。注意，右双引号前面有一个空格，这让程序知道我们希望最后一个单词后面有一个空格。Java 会忽略大部分空白，但空格位于用引号括起的文本字符串中时，它将是文本字符串的一部分。

我们还要打印用户猜测的数字。这个值存储在变量 guess 中，因此需要使用 println()语句来输出它。所幸在 Java 中，当你在 println()语句中包含变量时，Java 将打印变量的值。因此，我们紧跟在文本"You entered "后面添加了拼接运算符（+）和变量名 guess。最后，我们想在句子末尾加上句点，因此再添加一个拼接运算符，并在它后面加上要打印的文本，再将其用双引号括起（"."）。

代码清单 2-3 列出了我们到目前为止编写的所有代码行。

代码清单 2-3　生成一个随机数并让用户猜测一次的代码

```
import java.util.Scanner;

public class HiLo {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        // 生成一个要让用户去猜的随机数
        int theNumber = (int)(Math.random() * 100 + 1);
      ❶ // System.out.println( theNumber );
        int guess = 0;
        System.out.println("Guess a number between 1 and 100:");
        guess = scan.nextInt();
        System.out.println("You entered " + guess + ".");
    }
}
```

请注意，在❶处，我将代码行 System.out.println(theNumber);变成了注释，这是通过在行首添加两个斜杠实现的。这被称为**注释掉**，是一种很有用的**调试**（查找并修复程序中的 bug 或错误）方法。前面编写并测试程序时，我们使用这条 println()语句来显示变量 theNumber 的值，但现在我们要让计算机忽略它，为此我们没有将这行代码删除，而将其变成了注释。如果以后又想使用这行代码，只需将//删除就可将它包含在程序中。

下面保存并运行这个程序，看看它能否正确地工作。要运行程序，可单击绿色的 Run 按钮，也可选择菜单 Run▶Run。当前，用户只能猜一次，且程序没有检查猜测是否正确。因此，接下来将添加一些代码，让用户能够猜测多次。同时还将介绍如何检查猜测的数字是否与 theNumber 相同。

2.4　循环：反复地询问并检查

要给用户多次猜数机会，我们需要学习如何创建循环。在这个猜数游戏中，我们需要不断地让用户猜测，直到猜对为止。**循环**让我们能够反复地执行一系列步骤。本节将创建一个循环，在其中让用户猜测并获取键盘输入。

循环是功能极其强大的编程工具，这也是计算机在日常生活和商业领域中如此有价值的原因之一。计算机确实很擅长执行重复的任务，只要程序没有问题，计算机就能整天执行重复的任务且不会出错。反复跟用户说他猜测的数字太大或太小时，你我可能会厌烦，但计算机绝不会厌烦。它也不会忘记要猜的数字是多少，更不会在指出用户猜测的数字太大或太小时出错。

下面就来利用循环的强大威力；我们将使用的是 while 循环。while 循环在**条件**满足时反复执行一组语句。条件是可检查的情况，例如，在这个程序中，我们要知道用户是否猜对了。如果没有猜对，我们就再给用户一次机会，直到他猜对为止。

要编写 while 循环，需要知道每次执行循环前需要检查的条件。在这个猜数游戏中，只要用户的猜测与秘密数字 theNumber 不相等，我们就让他再猜一次。用户的猜测与秘密数字相等后，用户将获胜，而游戏也将结束，因此循环应就此终止。

为创建 while 循环，我们需要在最后 3 行代码前面插入一条 while 语句，并将这 3 行代码放在一对大括号内，如下所示：

```
        int guess = 0;
        while (guess != theNumber) {
        System.out.println("Guess a number between 1 and 100:");
        guess = scan.nextInt();
        System.out.println("You entered " + guess + ".");
        }
    }
}
```

我们使用关键字 while 告诉 Java 我们要创建一个 while 循环，然后我们将合适的条件放在括号内。括号中的内容 guess != theNumber 意味着只要 guess 的值与 theNumber 的值不相等（!=），就执行这行代码后面的语句。!=是一个**比较运算符**；在这里，它比较 guess 和 theNumber，并指出它们是否不同（**不相等**）。就这个让用户猜测的 while 循环而言，只需要这个比较运算符，其他比较运算符将在下一节介绍。

我们需要让 Java 知道要反复执行哪些语句，因此我们在 while 语句后面添加了一个左大括号（{）。在 main()方法中，大括号将所有语句编组。同样，这里的大括号将 while 循环中的语句编组。

我们要在这个循环中包含 3 条语句。首先，我们需要使用 println()语句来提示用户猜数；接下来，需要使用方法 nextInt()获取并记录用户的猜测。最后，需要使用 println()语句告诉用户他猜测的数字是多少。为将这些语句作为供 while 语句反复执行的代码块，我们首先编写 while 语句和条件，再依次加上左大括号、这 3 条语句和右大括号。千万别忘了右大括号，否则程序将无法运行。

一种不错的做法是使用制表符来缩进，这可让代码组织有序且可读性强。为此，请选择位于 while 语句的大括号内的 3 条语句，再按 Tab 键来缩进它们。

缩进后的代码应类似于下面这样：

```
        int guess = 0;
        while (guess != theNumber) {
            System.out.println("Guess a number between 1 and 100:");
            guess = scan.nextInt();
            System.out.println("You entered " + guess + ".");
        }
    }
}
```

正确的缩进有助于你记着给左大括号加上配套的右大括号，还有助于快速看清哪些语句位于循环或代码块内、哪些语句在循环外。缩进对程序运行没有任何影响，但正确地缩进可让程序更容易理解和维护。

将程序存盘并运行，看看它能否正确地工作。这个游戏差不多能玩了，但我们还需要让它检查用户猜大了、猜小了还是猜对了。现在该 if 语句出场了（请奏乐）!

2.4.1　if 语句：检查合适的条件

让用户能够不断地猜测，直到猜对为止后，需要检查用户的猜测，让他知道是猜大了还是猜小了，为此可使用 if 语句。

if 语句根据条件（**条件表达式**）决定是否执行指定的语句块。

在前面让用户不断猜测的循环中，我们使用了一个条件表达式：(guess != theNumber)。要检查猜大了还是猜小了，需要使用另外几个比较运算符：小于（<）、大于（>）和等于（==）。

首先，我们编写一些代码来检查用户猜测的数字是否太小了，而不指出用户猜测的数字是什么。为此，请将 while 循环中的最后一行替换为下面这条包含两行的 if 语句：

```
        while (guess != theNumber) {
            System.out.println("Guess a number between 1 and 100:");
            guess = scan.nextInt();
            if (guess < theNumber)
                System.out.println(guess + " is too low. Try again.");
        }
```

if 语句以关键字 if 打头，接下来是用括号括起的条件表达式。在这里，条件为 guess < theNumber，这意味着用户猜测的数字小于生成的随机数。请注意，括号后面没有分号，因为接下来的 println()

语句实际上是这条 if 语句的一部分。整个 if 语句告诉程序，如果满足条件，就打印用户猜测的数字，并让他知道猜小了。在用户猜测的数字和告诉用户猜小了的文本字符串之间，我们使用了拼接运算符（+）。请注意，第一个双引号和 is 之间有一个空格，这样将把用户猜测的数字和单词 is 分开。

如果你现在运行这个程序，并输入比生成的随机数小的数字（如 1），这条 if 语句将让程序指出你猜小了。这是一个良好的开端，但如果用户猜测的数字比生成的随机数大呢？为处理这种情况，需要一条 else 语句。

else 语句让程序在 if 语句中的条件不满足时走另一条路径（即执行另一组操作）。通过使用一对 if-else 语句，我们可以检查用户猜大了还是猜小了。下面在 if 语句后面添加一条 else 语句：

```
❶ if (guess < theNumber)
      System.out.println(guess + " is too low. Try again.");
❷ else if (guess > theNumber)
      System.out.println(guess + " is too high. Try again.");
```

请注意，❷处的代码与❶处的代码类似。使用 if-else 语句时，通常需要依次检查多个条件（而不是只检查一个条件）。在这里，我们需要检查用户猜大了、猜小了还是猜对了。在这种情况下，可以串接 if-else 条件：将下一条 if 语句放在前一条 if-else 语句的 else 部分中。在❷处，我们在前一条 if 语句的 else 后面立即开启了下一条 if 语句：如果 guess 大于 theNumber，程序将告诉用户猜大了。至此，程序能够告诉用户猜大了还是猜小了，我们只需再告诉用户是否猜对并获胜了即可！

如果前面两个条件都不满足（既没有猜大也没有猜小），那么用户必然是猜对了。因此，我们添加最后一条 else 语句：

```
❶ if (guess < theNumber)
      System.out.println(guess + " is too low. Try again.");
❷ else if (guess > theNumber)
      System.out.println(guess + " is too high. Try again.");
❸ else
      System.out.println(guess + " is correct. You win!");
```

请注意，在最后一条 else 语句❸中，不需要条件表达式。如果既没有猜大也没有猜小，就只能是猜对了。为处理用户猜对了的情形，我们使用一条语句让用户知道他赢了。至此，完整的程序代码如代码清单 2-4 所示。请将文件 HiLo.java 存盘，并运行这个程序，看看它能否正确地工作。这个程序应让你不断地猜测，直到猜对为止。

代码清单 2-4 让用户能够玩一轮的猜数游戏

```
import java.util.Scanner;

public class HiLo {
    public static void main(String[] args) {
```

```
            Scanner scan = new Scanner(System.in);
            // 生成一个要让用户去猜的随机数
            int theNumber = (int)(Math.random() * 100 + 1);
            // System.out.println( theNumber );
            int guess = 0;
            while (guess != theNumber) {
                System.out.println("Guess a number between 1 and 100:");
                guess = scan.nextInt();
                if (guess < theNumber)
                    System.out.println(guess + " is too low. Try again.");
                else if (guess > theNumber)
                    System.out.println(guess + " is too high. Try again.");
                else
                    System.out.println(guess + " is correct. You win!");
             } // 猜数循环到此结束
        }
    }
```

现在，这个程序是一个完整的猜数游戏了！用户猜对后，这个程序将指出他猜对并赢了，然后结束，如图 2-6 所示。

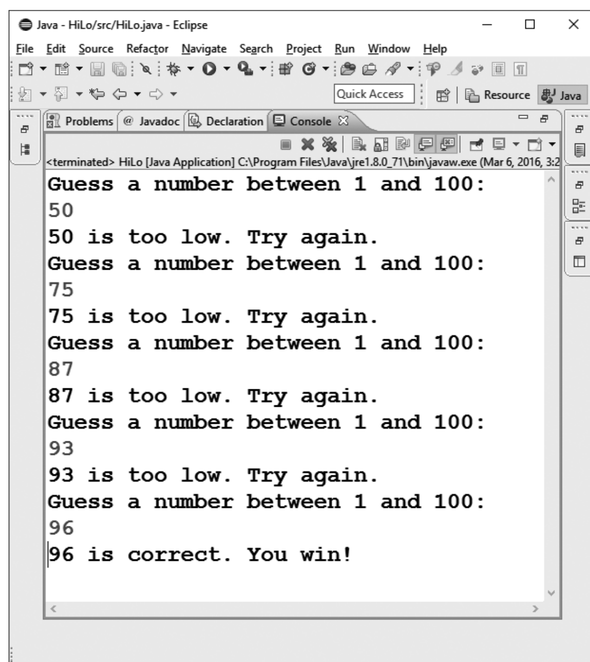


图 2-6　可玩一轮的猜数游戏——程序在用户猜对后结束

给自己鼓鼓掌吧！你从头创建了一个 Java 程序，如果这是你使用 Java 编写的第一个程序，绝对值得庆贺。请玩这款游戏几次，看看你猜测的次数是否一次比一次少。对这个程序进行测试，确保它像你期望的那样工作，下一节将对其做些改进。

2.4.2　添加让用户接着玩的循环

当前，要再次玩这个猜数游戏，只能在 Eclipse 中再次运行它。所幸我们已经知道一种让程序反复执行操作的办法——再添加一个循环！

这个猜数游戏在用户猜对后结束，因为 while 循环后面什么都没有。这个 while 循环在条件 (guess != theNumber)不再满足时结束。掌握窍门后，用户可能想反复地玩。为编写这个让用户能够反复玩的循环，我们将学习一个新的关键字，还有一种新循环——do-while 循环。

与 while 循环一样，do-while 也在条件满足时反复地执行语句块；但与 while 循环不同的是，do-while 循环中的代码块至少会执行一次。在有些情况下，while 开头的条件在循环还未开始时就可能不满足，因此整个循环以及其中的所有代码行都将被忽略。While 循环中的条件犹如加热器的温度调节器。如果房间的温度足够高，高到都不满足启动加热器的条件，加热器根本就不会启动。

对于这里的猜数游戏以及几乎所有的游戏程序，我们都选择使用 do-while 循环（我们有时称之为**游戏循环**），因为用户可能想至少玩一次游戏。我们通常还会询问用户是否想再玩，而用户通常以 yes 或 no（在基于文本的游戏中为 y 或 n）作答。只要用户以 yes 作答，游戏循环就不会终止。

为检查用户的回答，需要一个类型为 String 的变量。String 是一种对象，存储用双引号括起的文本，如"y"、"yes"或"My name is Bryson! I hope you like my game!"。在本章前面，我们使用了一个整型（类型为 int 的）变量来存储用户猜测的数字，但现在我们需要存储文本，因此将使用 String 变量。我们可在程序开头（紧跟在创建 Scanner 对象的代码后面）添加一个 String 变量：

```
Scanner scan = new Scanner(System.in);
String playAgain = "";
```

注意，类型名 String 的首字母是大写的，这是因为 String 实际上是一个类，包含多个可用于处理文本字符串的函数。我将这个变量命名为 playAgain，这使用了骆驼拼写法，即将第二个单词的首字母大写（A）。别忘了，变量名中不能有空格。前面使用 int guess = 0 将变量 guess 的初始值设置成了 0，同样，这里也使用 playAgain = ""给变量 playAgain 指定了初始值。两个双引号（中间没有空格）表示空字符串，即这个 String 变量不包含任何文本。后面在用户输入 y 或 n 时，我们将把不同的文本值赋给这个变量。

与前面编写 while 循环一样，我们需要确定哪些语句是要在 do-while 循环中反复执行的。这个 do-while 循环为主循环，因此程序的几乎所有语句都将放在其中。事实上，除创建 Scanner 对象和 String 变量 playAgain 的语句外，其他所有语句都将包含在这个 do-while 循环中。这些语句是一整轮游戏包含的步骤，因此在每一轮中，游戏都将重复所有这些步骤：从生成新的随机数，到宣布用户猜对了并询问用户是否还要接着玩。

因此，我们可以在前述两行代码和生成随机数的代码之间添加关键字 do 和左大括号：

```
Scanner scan = new Scanner(System.in);
String playAgain = "";
do {
    // Create a random number for the user to guess
    int theNumber = (int)(Math.random() * 100 + 1);
    // 生成一个要让用户去猜的随机数
    int guess = 0;
    while (guess != theNumber) {
```

接下来，在让用户不断猜数的 while 循环的右大括号（即最后一条 else 语句后面的右大括号）后面，我们询问用户是否要接着玩并获取他通过键盘做出的回答。

然后，我们需要指定 do-while 循环的结束位置，为此可使用一个检查用户的回答是否为 yes 的 while 条件：

```
        } // 让用户不断去猜的while循环到此结束
      ❶ System.out.println("Would you like to play again (y/n)?");
      ❷ playAgain = scan.next();
    ❸ } while (playAgain.equalsIgnoreCase("y"));
  ❹ }
❺ }
```

这里使用了提示语"Would you like to play again（y/n）?" ❶，用户可使用单个字符来做出回答：表示 yes 的 y 或表示 no 的 n。在❷处，函数 scan.next() 从键盘获取输入，但它不像 nextInt() 那样获取下一个整数，而是获取用户通过键盘输入的一个或多个字符。不管用户输入的是什么，都将存储到变量 playAgain 中。

❸处的代码行使用大括号指定反复执行游戏的代码块的结束位置，其中还包含确定是否要再次执行这些代码的 while 条件。在这个 while 条件中，使用了 String 对象的方法 equalsIgnoreCase()，它确定两个字符串在不区分大小写的情况下是否相等，这不同于在比较字符串时区分大小写的方法 equals()。在这个游戏中，我们让用户输入 y 来表示要接着玩，但如果我们只检查小写 y，就会遗漏大写 Y。在这里，我们想提高灵活性，检查用户输入是否是小写 y 或大写 Y，因此使用了字符串方法 equalsIgnoreCase()。

最后的 while 语句告诉 Java，只要字符串变量 playAgain 为小写 y 或大写 Y，就接着执行游戏循环。最后两个右大括号（❹和❺处）是早就有的；其中❹处的右大括号指出 main() 方法到此结束，而❺处的右大括号指出整个 HiLo 类到此结束。这里列出它们旨在指出应将❶到❸处的代码行插入到什么地方。

到目前为止，整个游戏如代码清单 2-5 所示。

代码清单 2-5　现在猜数游戏可反复地玩了

```
import java.util.Scanner;
public class HiLo {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        String playAgain = "";
```

```
        do {
            // 生成一个要让用户去猜的随机数
            int theNumber = (int)(Math.random() * 100 + 1);
            // System.out.println( theNumber );
            int guess = 0;
            while (guess != theNumber) {
                System.out.println("Guess a number between 1 and 100:");
                guess = scan.nextInt();
                if (guess < theNumber)
                    System.out.println(guess + " is too low. Try again.");
                else if (guess > theNumber)
                    System.out.println(guess + " is too high. Try again.");
                else
                    System.out.println(guess + " is correct. You win!");
              } // 让用户不断去猜的while循环到此结束
              System.out.println("Would you like to play again (y/n)?");
              playAgain = scan.next();
          } while (playAgain.equalsIgnoreCase("y"));
      }
}
```

请审核你的代码，确保所有代码都位于正确的地方，同时检查大括号和分号，并将文件存盘。下一节将对这个游戏进行测试。

注意 在你的代码中，缩进（每行开头的制表符）可能与这里显示的不完全相同，因为我们在两个地方新增了大括号。所幸在 Java 中经常需要添加新功能，包括循环和其他代码块，因此 Eclipse 提供了一个自动整理缩进的菜单项。首先，请选择文件 HiLo.java 中的所有代码，再选择菜单 Source▸Correct Indentation，Eclipse 将正确地缩进每行代码，以指出哪些语句属于同一个代码块。前面说过，缩进对计算机来说毫无意义（即便没有任何制表符和额外的空格，程序也能正确运行），但良好的缩进和间隔让程序更容易理解。

2.5 测试游戏

添加让用户能够接着玩的循环后，这个游戏运行起来应该非常完美。首先，将文件 HiLo.java 存盘，再选择 Run▸Run 对这个游戏进行测试。当你猜对第一个随机数后，游戏将询问你是否要接着玩。只要你输入 y（或 Y）并按回车，游戏就会再次生成随机数并让你去猜。从图 2-7 所示的屏幕截图可知，当游戏询问我是否要接着玩，而我以 y 作答时，游戏又重新开始了。

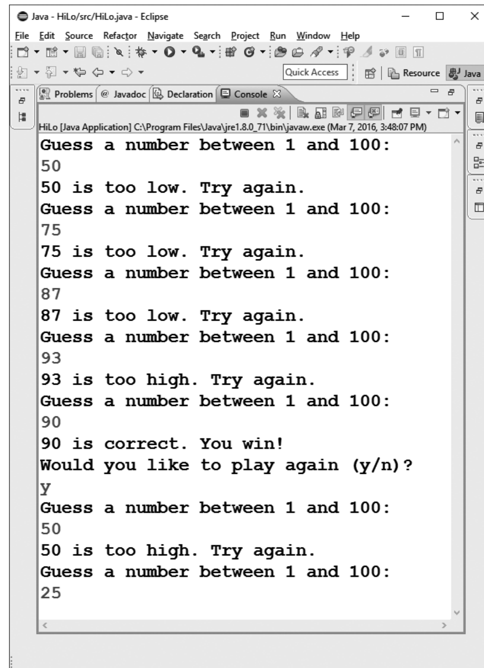


图 2-7　只要以 y 或 Y 作答，用户就能玩猜数游戏多次

用户不想再玩，进而以 n（或除 y 和 Y 之外的其他字符）作答时，游戏将结束。然而，你可能想在用户结束游戏时表示感谢。为此，可在最后一条 while 语句和最后两个右大括号之间添加如下代码行：

```
        } while (playAgain.equalsIgnoreCase("y"));
        System.out.println("Thank you for playing! Goodbye.");
    }
}
```

最后，我们还需在这个猜数游戏中添加一行代码，以消除 Eclipse 发出的警告。你可能注意到了，这个警告表现为 scan 对象声明下方的淡黄色线条，还有这行代码左边的带惊叹号的黄色三角形。Eclipse 这样做旨在让我们注意，我们打开了一个资源，却没有关闭它。在编程中，这可能引发**资源泄露**。如果只打开了一个 Scanner 对象以获取键盘输入，这通常无关紧要，但如果打开了多个 Scanner 对象，却没有关闭它们，程序可能耗尽内存，导致用户的系统变慢甚至崩溃。要让程序关闭到键盘的连接，可调用 Scanner 类的方法 close()。

请在感谢用户的 println()语句后面（最后两个右大括号前面）添加如下代码行：

```
        System.out.println("Thank you for playing! Goodbye.");
        scan.close();
    }
}
```

添加这行代码后，你将发现 Eclipse 编辑器窗口中的黄色警告消失了。Eclipse 能够发现常见的编程错误，如拼写错误和标点缺失，它还能就可能存在的问题（如资源泄露和变量未被使用）发出警告。使用 Java 创建更大、更复杂的应用程序时，Eclipse 的这些功能将显得更为弥足金贵。有关如何使用 Eclipse 来调试程序的更详细信息，请参阅附录。

最终的程序如代码清单 2-6 所示，这是一个可玩性很高的猜数游戏，它会询问用户是否要接着玩，并在每轮中都重新生成一个随机数。

代码清单 2-6 最终的基于文本的命令行猜数游戏

```
import java.util.Scanner;

public class HiLo {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        String playAgain = "";
        do {
            // 生成一个要让用户去猜的随机数
            int theNumber = (int)(Math.random() * 100 + 1);
            // System.out.println( theNumber );
            int guess = 0;
            while (guess != theNumber) {
                System.out.println("Guess a number between 1 and 100:");
                guess = scan.nextInt();
                if (guess < theNumber)
                    System.out.println(guess + " is too low. Try again.");
                else if (guess > theNumber)
                    System.out.println(guess + " is too high. Try again.");
                else
                    System.out.println(guess + " is correct. You win!");
            } // 让用户不断去猜的while循环到此结束
                System.out.println("Would you like to play again (y/n)?");
                playAgain = scan.next();
            } while (playAgain.equalsIgnoreCase("y"));
            System.out.println("Thank you for playing! Goodbye.");
            scan.close();
    }
}
```

有关这个最终的猜数游戏，有几点需要说说。首先，虽然为编写这款游戏需要完成的工作量很多，但其代码相对较短，总共不超过 30 行。尽管如此，如果你愿意，这款游戏可没完没了地玩下去。其次，这个程序不仅演示了条件和循环，还在一个循环中使用了另一个循环。这被称为**嵌套循环**，因为猜数循环包含（嵌套）在重玩循环中。缩进有助于我们看清 do-while 循环从哪里开始、到哪里结束；我们注意到，较小的 while 循环及其 if 语句缩进了一个制表位，并嵌套在较大的 do-while 循环中。最后，我们优雅地终止程序，对用户（感谢用户玩这款游戏）和计算机（关闭 Scanner 对象）来说都如此。

2.6　小结

本章创建了一个有趣又好玩的简单游戏，在此过程中我们学习了多个重要的编程概念。这正是我在孩童时期学习编程的方式：找一个有趣的游戏或图形应用程序，将其编写出来，再进行修改、拆解以及尝试添加新功能。无论学习什么新知识，把玩和探索都是学习过程中的重要组成部分，希望你花点时间尝试在每个程序中添加新功能。本书每章末尾都有编程练习，让你有机会去尝试一些新鲜事物。

在编写这个猜数游戏的过程中，你学会了很多 Java 编程技能：

- 创建新类（HiLo）；
- 导入既有的 Java 包（java.util.Scanner）；
- 使用 Scanner 对象获取键盘输入；
- 声明并初始化整型变量和字符串变量；
- 使用 Math.random()生成随机数并将其强制转换为整数；
- 使用 while 和 do-while 循环在满足条件的情况下反复执行一系列步骤；
- 将文本字符串和变量的值打印到命令行控制台；
- 从键盘获取整数和字符串并将其存储到变量中；
- 在 if 和 if-else 语句中检查各种条件表达式；
- 使用 String 类的方法 equalsIgnoreCase()来比较字符串值；
- 使用方法 close()来关闭诸如 Scanner 对象等输入资源；
- 在 Eclipse 中运行命令行程序。

除这些实用技能外，你还学习了多个重要的 Java 编程概念。

- **变量**：theNumber 是一个整型（int）变量，guess 亦如此。playAgain 是一个字符串（String）变量。在玩游戏的过程中，用户通过输入猜测的数字以及以 y 或 n 作答来修改这些变量的值。
- **方法**：在 Java 中，方法是类中的函数。方法 Math.random()用于生成 0.0~1.0 的随机数；方法 scan.nextInt()接受来自用户的数值输入；System.out.println()向控制台或终端窗口显示文本。
- **条件**：if-else 让你能够检查条件（如 guess < theNumber）是否满足，并根据检查结果执行不同的代码块。你还可使用条件表达式来决定是否再次执行循环，如 while (guess != theNumber)；就这条语句而言，只要 guess 不等于 theNumber，就将不断执行循环。别忘了，两个等号（==）用于检查相等性。
- **循环**：while 循环让我们能够在条件满足的情况下反复执行代码块。在猜数游戏中，我们使用一个 while 循环让用户不断地猜测，直到猜对为止。do-while 循环至少会执行一次，我们使用了一个这样的循环来询问用户是否要接着玩。
- **类**：整个 HiLo 应用程序就是一个 Java 类——公有类 HiLo。类就是模板。本章为猜数游戏创建了一个类模板，可在众多不同的计算机中使用它来玩猜数游戏。在这个应用程序

中，我们还导入了 Scanner 和 Math 类，以便使用它们来获取用户输入以及生成随机数。我们编写自定义类来实现新功能，并利用 Java 提供的类来完成诸如输入、数学计算等日常任务。

2.7 编程练习

为复习并使用学到的知识，以及获得更多的编程技能，请尝试完成这里的编程练习。如果遇到困难，可从本书的配套网站（https://www.nostarch.com/learnjava/）下载示例解决方案，也可前往 http://www.udemy.com/java-the-easy-way/观看视频课程，其中提供了详尽的解决方案。本章的视频课程可免费观看，如果你要购买完成的视频课程，使用优惠码 BOOKHALFOFF 可打 5 折。

2.7.1 编程练习 1：增大范围

在这个编程练习中，请修改猜数游戏，从更大的范围内随机选择一个数字。不让用户猜 1~100 的数字，而让他猜–100~100 的数字。

提示 要生成这样的随机数，可将 Math.random()乘以 200 再减去 100。

在修改生成随机数的语句的同时，别忘了修改提示，将猜测范围告诉用户。

如果你希望这个游戏更容易，可将范围修改为 1~10，并在你能够在 4 次内猜对时向朋友显摆一下。尝试其他范围，如 1~1000 甚至 1~1 000 000。如果你愿意，也可尝试负数范围。（记住：用 Java 写数字时，不能使用逗号。）通过这个练习，你不仅会更擅长编程，数学技能也将得到提高。可随心所欲地修改这个程序，祝你玩得愉快！

制定猜数策略

你可能发现，玩猜数游戏的次数越多，你猜对的速度越快。你甚至可能无意间发现这样一个规律：如果每次都猜范围的中间值，可以最快的速度猜对数字。这种方法被称为**二分查找**。通过猜可能范围的中间值，可每次将可能的数字减少一半。

下面来说说具体做法。对于 1~100 的数字，首先猜 50。如果太小，秘密数字必然在 51 到 100 之间，因此你接着猜这个范围的中间位置——75。如果也太小，就接着猜 76 到 100 的中间值——87。二分查找很有用，其中一个原因是对于 1~100 的数字，最多只需猜 7 次就能猜对。你可以试一试！

掌握最多 7 次就能猜对 1~100 的数字的窍门后，请尝试在 10 次内猜对 1~1000 的数字。如果你勇气爆棚（或者身边有支铅笔），可尝试去猜一个 1~1 000 000 的数字。信不信由你，你在 20 次内就能猜对。

2.7.2　编程练习 2：计算猜测次数

本章创建的猜数游戏很酷，但请你尝试再添加一项功能：计算并指出用户猜了多少次才猜对。需要打印的输出类似于下面这样：

```
62 is correct! You win!
It only took you 7 tries! Good work!
```

为完成这项任务，需要创建一个新的变量（你可能添加类似于 `int numberOfTries = 0;`的代码行）。你还需在猜数循环每次执行时都将猜测次数加 1，为此可在这个循环中使用 `numberOfTries = numberOfTries + 1` 将变量 `numberOfTries` 加 1。请务必包含让用户知道猜测次数的文本。

为编写出正确的代码，并让它们在正确的时间按正确的顺序执行，你可能需要尝试多次。但这样做是值得的，因为你将把新学到的技能付诸实践。在第 3 章中，我们将创建另一个版本的猜数游戏，并在其中添加这项功能。届时，但愿你能想出更多改进和修改这款游戏的点子。请把玩你编写的程序，将其拆解再组装起来，这是最佳的学习方式。

2.7.3　编程练习 3：玩 MadLibs 游戏

在本章的最后一个编程练习中，我们来编写一个全新的程序。你学习了如何请求用户提供输入并将其存储到变量中，还学习了如何将文本和变量的值打印到屏幕上。利用这些技能，可创建更有趣、更好玩的程序。

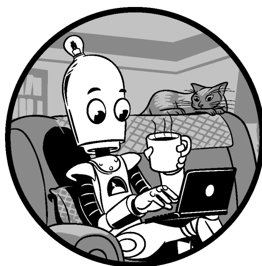
玩过 MadLibs 游戏吗？我们来尝试使用新学的技能创建一个这种风格的程序。MadLibs 让玩家输入各种单词或短语，如颜色、动词的过去时或形容词，再将用户选择的单词插入一个模板，这通常会得到一个有趣的故事。例如，如果玩家输入颜色 pink、动词过去时 burped 和形容词 silly，并将它们插入模板“The ____ dragon ____ at the ____ knight”，结果将为“The *pink* dragon *burped* at the *silly* knight”。

现在，你需要做的是编写一个新程序——MadLibs.java，它包含一个名为 `MadLibs` 的类，而这个类的方法 `main()`提示用户输入多个单词。这些单词应存储在不同的 `String` 变量中，如 `color`、`pastTenseVerb`、`adjective` 和 `noun`，而这些变量被初始化为空字符串。接下来，在用户输入最后一个单词后，这个程序应将空字符串替换为用户输入的单词，并打印得到的句子或故事，如下所示：

```
System.out.print("The " + color + " dragon " + pastTenseVerb + " at the " + adjective);
System.out.println(" knight, who rode in on a sturdy, giant " + noun + ".");
```

请注意，第一条语句使用的是 `print()`而不是 `println()`。`print()`打印后不换行，让我们能够创建更长的段落或故事；而 `println()`总是在打印后换行，就像你在行尾按回车一样。要创建更长的 MadLibs 故事，可使用不同的变量名，如 `noun1`、`noun2` 和 `noun3`。请尝试这样做一做，看到你创建的有趣的故事时，可别笑破肚皮哟！对于你创建的每个程序，尝试对其进行定制：添加新的功能，让它变成你自己的。

第 3 章 给猜数游戏创建 GUI



本章将创建第 2 章编写的猜数游戏的 GUI 版本,如图 3-1 所示。这个版本的程序在运行时显示一个用户将与之交互的图形界面。这个 GUI 版本就像你每天使用的应用程序一样,让用户能够在窗口中玩游戏。这是一个专业的窗口式应用程序，包含让用户输入猜测的文本框、提交猜测的按钮，以及指出用户猜大了、猜小了或猜对了的标签。

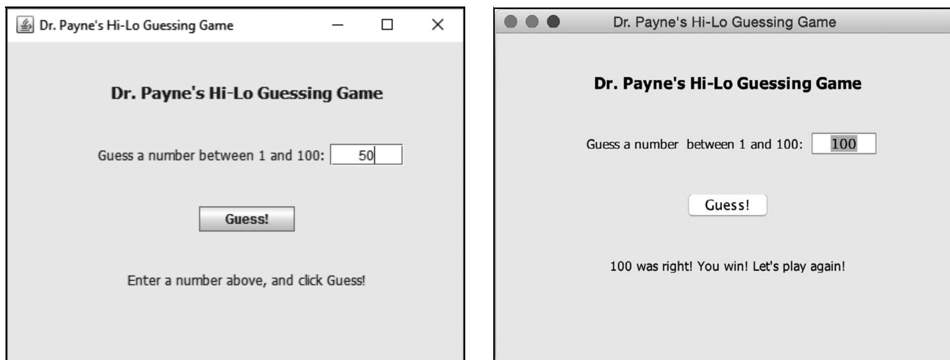


图 3-1　在 Windows（左）和 MacOS（右）系统中运行的 GUI 版猜数游戏

最重要的是，图 3-1 所示的两个版本的应用程序是由同样的 Java 代码生成的。有了本章编写的 GUI 代码后，无论用户使用的操作系统是 Windows、macOS 还是 Linux，都可玩这个猜数游戏！

3.1　在 JShell 中练手

JShell 运行在命令行窗口中，它接受文本命令且通常输出文本。然而，JShell 还能够访问一系列 Java 库（预先编写好的代码包），而不仅仅是文本。着手开发本章的游戏前，我们先来练练手——在 JShell 中创建一个简单的 GUI。

3.1.1　仅用 4 行代码创建一个 GUI

在 JShell 中，只需 4 行代码就能创建一个简单的 GUI 窗口。我们每次看一条语句。首先，在 JShell 中输入一条 import 语句，它导入 javax.swing.JFrame 类：

```
jshell> import javax.swing.JFrame
```

刚才导入的 JFrame 类使用 Java 来创建框架（窗口）。接下来，我们使用这个类创建一个框架。为此请输入如下声明，创建一个名为 myFrame 的 JFrame：

```
jshell> JFrame myFrame = new JFrame("Hello!")
```

关键字 new 新建一个 JFrame 对象，这里是一个在标题栏中显示"Hello!"的 GUI 窗口。JShell 以一行很长的内容进行响应，让我们知道这个窗口包含的一些默认属性：

```
myFrame ==> javax.swing.JFrame[frame1,0,0,0x0,invalid,hidden,layout=java.awt.BorderLayou ...
```

方括号内的信息是 myFrame 及其属性值的字符串表示，这些属性包括尺寸（0x0，表示 0 像素 × 0 像素）以及是隐藏还是可见的等。下面来修改一个属性：让尺寸不再为 0 像素 × 0 像素。

下面是第 3 行代码，它以像素为单位设置窗口的尺寸，这是通过指定框架的宽度和高度实现的：

```
jshell> myFrame.setSize(300,200)
```

这行代码让 Java 将窗口的宽度和高度分别设置为 300 像素和 200 像素，这足以让我们看到窗口和标题栏。

最后，调用方法 setVisible()在屏幕上显示这个窗口：

```
jshell> myFrame.setVisible(true)
```

要显示窗口，可使用 setVisible(true)；要隐藏它，可使用 setVisible(false)。由于调用了 myFrame.setVisible(true)，你应该在屏幕上看到窗口，如图 3-2 所示。

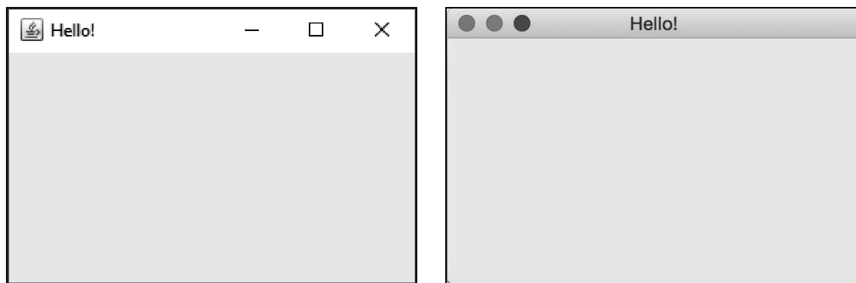


图 3-2　Windows（左）和 macOS（右）系统中的 GUI 窗口

在 JShell 中，我们只输入 4 行代码就创建了一个 GUI 窗口。

3.1.2　用 10 行代码创建一个交互式 GUI

我在大学向学生演示过前面的示例，也在家里向儿子演示过它，他们都说："这真的很酷呢！但你能让这个窗口做点事情吗？"

谢天谢地，答案是肯定的。只需再添加几行代码，就可在这个窗口中添加一个按钮，而当你单击这个按钮时，它将打印一些内容。

我们重新开始一个 JShell 会话。为此，在 JShell 提示符下输入`/reset`并按回车，以清除 JShell 历史记录：

```
jshell> /reset
| Resetting state.
```

接下来，我们使用内置的 JShell 编辑器编写一个 10 行的代码片段，以创建一个可响应单击的交互式 GUI 应用程序。请在 JShell 提示符下输入`/edit`并按回车，以打开 JShell 编辑器：

```
jshell> /edit
```

这将打开 JShell Edit Pad，其中包含一个空窗口，如图 3-3 所示。在你需要一次性编写多行代码，或要回过头去编辑以前输入的代码行时，JShell Edit Pad 很有用。



图 3-3　JShell Edit Pad 是一个方便的编辑器，可用于编写较长的代码片段

在这里，我们要输入 10 行代码，这些代码将创建一个交互式 GUI 窗口，其中包含一个被单击时将打印内容的按钮。请输入下面的代码行。输入时要特别注意大小写，并在每条语句末尾加上分号，以将不同的命令分开：

```
  import javax.swing.*;
  JFrame window = new JFrame("Bryson's Window");
❶ JPanel panel = new JPanel();
  JButton button = new JButton(❷"Click me!");
❸ panel.add(button);
❹ window.add(panel);
  window.setSize(❺300,100);
❻ button.addActionListener(e ->
      System.out.println("Ouch! You clicked me!"));
❼ window.setVisible(true);
```

首先，我们导入了包括`JFrame`、`JPanel`和`JButton`在内的所有 Swing GUI 类。Java 库末尾的星号（*）被称为**通配符**，意思是包含指定包中的所有类。接下来，我们像前一个示例那样创建

了一个 JFrame。在❶处，我们在窗口内创建了一个面板（panel），这个面板将作为诸如标签和按钮等其他 GUI 组件的容器。

接下来，我们添加了一个显示文本“Click me!”的按钮❷。在❸处，我们将这个按钮添加到 GUI 面板中，再将这个面板添加到窗口中❹。然后，像前一个示例那样，我们将窗口的尺寸设置为宽 300 像素、高 100 像素❺。

❻处的代码行是奇迹发生的地方：我们给“Click me!”按钮添加了一个操作监听器，以便在用户单击这个按钮时做出响应。每当用户单击这个 GUI 按钮时，该操作监听器都将向控制台打印"Ouch! You clicked me!"。在本书创建的 GUI 应用程序中，都将使用类似的监听器，让程序在用户执行操作时同用户交互。

最后，我们让窗口可见❼，以便能够通过单击按钮来测试这个程序。

输入全部 10 行代码并核实它们准确无误后，单击 JShell Edit Pad 中的 Accept 按钮，以接受这些代码并在 JShell 中运行它们，如图 3-4 所示。

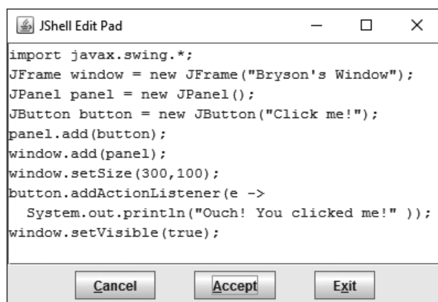


图 3-4　在 JShell Edit Pad 中输入全部 10 行代码后，单击 Accept

接受这些代码后，单击 Exit 关闭 JShell Edit Pad。JShell 将运行这个代码片段，如果正确地输入了所有的代码，将弹出一个类似于图 3-5 所示的小窗口。

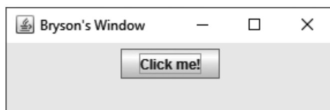


图 3-5　包含一个可单击按钮的交互式 GUI

单击名称为“Click me!”的按钮，Java 将在 JShell 窗口中显示如下内容：

```
jshell> Ouch! You clicked me!
Ouch! You clicked me!
Ouch! You clicked me!
Ouch! You clicked me!
```

关闭这个小型 GUI 窗口后，如果要重新显示它，只要再次执行让窗口可见的那行代码。为此，可在 JShell 命令行窗口中单击，再按回车两次，等提示符出现后输入如下代码并按回车：

```
jshell> window.setVisible(true)
```

这将 visible 属性设置为 true，因此窗口将再次出现；当你关闭窗口时，其 visible 属性将为 false。

这个 10 行的代码片段真的很酷，接下来你将学习如何创建带 GUI 的游戏。

3.2　在 Eclipse 中创建 GUI 应用程序

如果还没有在 Eclipse 中关闭第 2 章的项目 HiLo（如图 3-6 所示），请将你正在处理的所有 Java 文件的编辑器窗口关闭或最小化❶。另外，在左边的 Package Explorer 中，单击文件夹 HiLo 旁边指向下方的箭头❷，将项目 HiLo 折叠起来。这样可将项目放在同一个工作区，同时避免它们的文件混在一起。



图 3-6　关闭所有打开的文件并将项目文件夹 HiLo 折叠起来

在 Eclipse 中，选择菜单 File▸New▸Java Project，并将项目命名为 GuessingGame。输入项目名后，单击右下角的 Finish 按钮。这将新建一个项目，我们将在其中开发猜数游戏的 GUI 版本。

在 Package Explorer 中，展开文件夹 GuessingGame，并找到文件夹 src。右击（在 macOS 中是按住 control 并单击）文件夹 src，选择 New▸Class，并将新类命名为 GuessingGame。务必要使用骆驼拼写法，对于这个类名，还将首字母大写。选中 public static void main(String[] args)旁边的复选框，以添加方法 main()的骨架，这样我们就能将这个应用程序作为独立程序运行了。

最后，我们还需执行一个额外的步骤，这是第 2 章创建命令行应用程序时没有的。我们要将超类从默认的 java.lang.Object 改为 javax.swing.JFrame——就是本章开头在 JShell 中使用的 JFrame。在 Java 中，**超类**（**父类**）是我们为重用既有代码而要扩展的类，在这里，我们要重用的是创建窗口式图形界面所需的代码。要在应用程序中包含 GUI 组件，方法之一是使用 javax.swing 包中的 JFrame 类，因此我们将使用它来创建一个窗口，再通过定制来添加其他功能。JFrame 是超类 Object 的扩展，因此我们不需要 java.lang.Object，因为 JFrame **继承**了超类 Object 的代码。这意味着 JFrame 具备 Object 类的所有功能，还有其他一些功能。有关扩展的工作原理，将稍后介绍。

图 3-7 显示了创建 GuessingGame 类时需要在 New Java Class 对话框中所做的设置。



图 3-7　创建 GUI 类时，务必将超类改为 `javax.swing.JFrame`

单击 Finish 按钮，你将看到 GuessingGame.java 的源代码，如下所示：

```
❶ import javax.swing.JFrame;
public class GuessingGame ❷ extends JFrame {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
    }
}
```

在第 1 行，Eclipse 导入了 `javax.swing.JFrame` 类❶，让我们的程序能够使用 Swing 包提供的 GUI 功能。在第 2 行，我们使用了一种超强的编程功能——关键字 `extends`❷。

在面向对象编程中，父类（超类）可以有**子类**，而子类将继承父类的所有方法和属性。编写实现了重要功能的类后，要重用这些功能，可继承这个类并添加新功能（而无需修改这个类的代码）。在这里，父类 `JFrame` 提供了显示文本框、标签、按钮和其他 GUI 组件的功能，我们可根据猜数游戏的需求定制和排列这些组件。

在这个应用程序中，我们扩展了父类 `JFrame`，这让我们能够在 Eclipse 中使用 WindowBuilder Editor来设计GUI布局，从而显示包含文本框、标签和按钮等GUI元素的窗口。下面来看看 Window Builder Editor 的工作原理。

3.3　使用 Eclipse 的 WindowBuilder Editor 设计 GUI

大多数流行的 Java IDE 都提供了相关的工具，让程序员能够在设计视图中通过所见即所得

（WYSIWYG）界面创建引人入胜的 GUI。WYSIWYG 界面让用户能够定位设计元素，且看到的效果与最终产品中一样。例如，Microsoft Word 就是一种 WYSIWYG 界面，因为它让你能够编辑文本，且看到的效果与最终打印出来的效果相同。同理，Java IDE 提供的 WindowBuilder Editor 等工具让程序员能够定位文本框、标签和按钮等 GUI 组件，且看到的效果与最终的窗口式应用程序中一样，这可帮助程序员创建外观专业的 GUI 应用程序。

为了打开 WindowBuilder Editor，请在 Eclipse 窗口左边的 Package Explorer 中右击文件 GuessingGame.java，再选择 Open With▶WindowBuilder Editor。这将打开一个新窗口，它看起来像常规文本编辑器。但如果你看看这个窗口的左下角，将发现这里有两个选项卡：Source 和 Design。选项卡 Source 显示的是纯文本，即应用程序的源代码视图。如果你单击选项卡 Design，将看到如图 3-8 所示的设计视图。

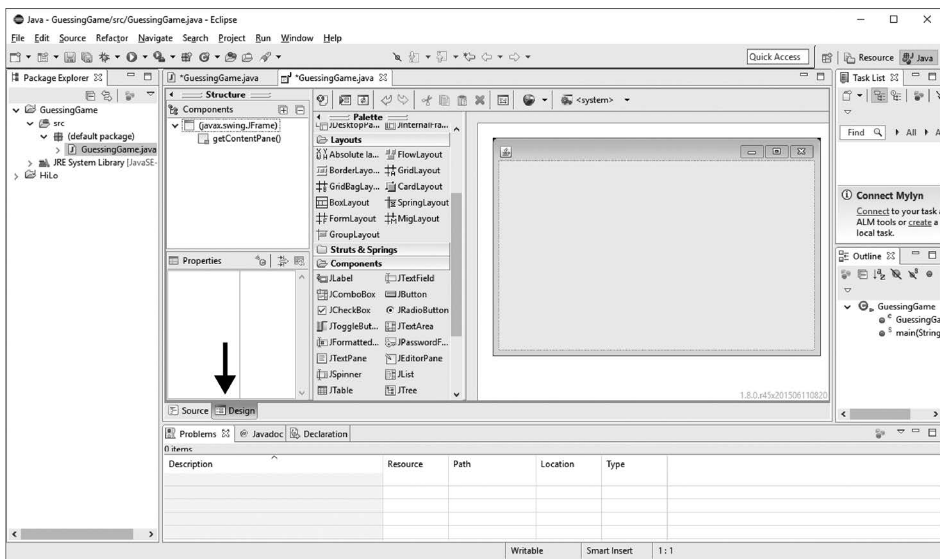


图 3-8 WindowBuilder Editor 的选项卡 Design 让你能够轻松地创建窗口式 GUI 应用程序

只有扩展了 `JFrame` 或其他 GUI 超类的 GUI 类（如 `GuessingGame`）才能在 WindowBuilder Editor 中打开。请注意，在预览窗口的标题栏中，左边有一个 Java 图标，而右边是三个按钮，它们分别用于将预览窗口最小化、最大化和关闭。在 Windows、macOS 和 Linux 系统中，预览窗口存在细微的差别。这个窗口将是我们创建 GUI 猜数游戏的操作台，它看起来像真的应用程序一样。

3.4 设计用户界面

请双击 WindowBuilder Editor 顶部的选项卡 GuessingGame.java，将 WindowBuilder 设计视图扩大到占满整个屏幕。这将为设计 GUI 布局腾出更多的空间。等你为回过头去编程做好准备后，可再次双击这个选项卡，以恢复到正常的 Java 透视图。

3.4.1　在 Properties 面板中设置 GUI 属性

我们将使用 Properties 面板来定制猜数游戏的 GUI 窗口。Properties 面板位于 WindowBuilder Editor 的 Design 选项卡的左下角。如图 3-9 所示，如果你单击 Components 面板（位于最左一列的 Structure 下方）中的组件，Properties 面板将列出该组件的多个属性及其值，我们可以查看并编辑它们。

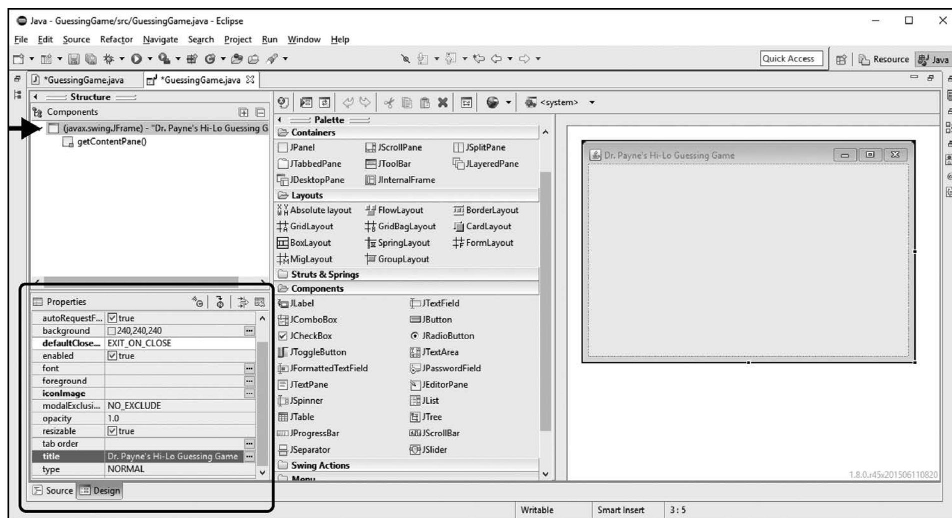


图 3-9　Properties 面板可帮助你对应用程序中每个 GUI 组件的属性进行定制

我们将为猜数游戏创建一个定制的 GUI 窗口。首先来修改这个 `JFrame` 的 `title` 属性。在 Components 面板中，单击 javax.swing.JFrame，再在 Properties 面板中向下滚动，直到看到属性 `title`。单击该属性旁边的空文本框，输入 *`Your Name`*`'s Hi-Lo Guessing Game`（请将 `Your Name` 替换为你的姓名，这毕竟是你的应用程序）并按回车。右边的 GUI 预览将自动更新，在标题栏中显示 *Your Name*'s Hi-Lo Guessing Game。

设置 `title` 属性后，再来设置其他几个属性，让 GUI 看起来像个猜数游戏。在 Properties 面板中向上滚动，以找到 `defaultCloseOperation`（如果这个面板太窄，可能只显示该属性名的一部分，如 `defaultClose...`）。单击该属性右边的方框，并从下拉列表中选择 EXIT_ON_CLOSE。这意味着当你关闭该 `JFrame` 的标题栏中的关闭窗口按钮时，将退出程序。这通常是默认行为，但在有些情况下，如在保存对话框或弹出窗口中，你可能想关闭窗口，而不是关闭整个应用程序。然而，对于本章的单窗口游戏，我们要在主窗口关闭时退出应用程序。

接下来，我们将修改这个应用程序窗口中 GUI 组件的排列方式。在 Components 面板中，单击 `javax.swing.JFrame` 下方的 `getContentPane()`。`JFrame` 内部是一个**内容面板**，我们将在其中设计猜数游戏的 GUI 布局。现在，在 Properties 面板中找到 Layout，单击它右边的下箭头，并从下拉列表中选择 Absolute Layout，这让我们能够精确地指定 GUI 元素的位置。

3.4.2 在 Palette 面板中定制 GUI 组件

下面来着手定制猜数游戏。不管是什么 GUI 应用程序，其所需的 GUI 元素几乎都可在 Palette 面板中找到，该面板位于 WindowBuilder Editor 的 Design 选项卡的中间。请注意，要展开/折叠面板，可单击其标题左边的小箭头：单击一次可将面板折叠起来，再次单击可重新展开它。在需要额外的空间以设计复杂的大型 GUI 布局时，这很方便。

在 Palette 面板中，向下滚动，直到能够看到 Components 部分，如图 3-10 所示。

3

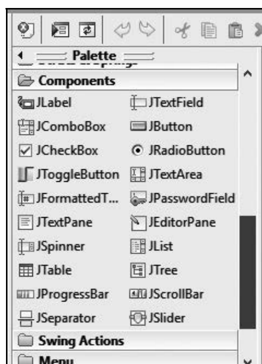


图 3-10 Palette 面板的 Components 部分包含最常用的 GUI 组件

Palette 面板包含所有标准组件，如标签（`JLabel`）、文本框（`JTextField`）、按钮（`JButton`）和复选框（`JCheckBox`），你可能对这些组件都很熟悉。你可能注意到了，`javax.swing` 包中的所有 GUI 组件都以大写字母 J 打头，后面跟着采用骆驼拼写法的组件名。这让你能够轻松地记住使用 Swing 工具包的应用程序的每个 GUI 元素的类名。

下面在 GUI 窗口顶部放置一个显示 *Your Name*'s Hi-Lo Guessing Game 的标签。为此，请单击 Palette 面板中 Components 部分的 JLabel，再将鼠标指向右边的 GUI 预览，你将看到网格线。默认情况下，WindowBuilder Editor 显示网格线来帮助你放置元素。请将鼠标指向灰色内容面板（`JFrame` 内部）的顶部中央附近并单击，在这里放置一个 `JLabel`。

刚添加 `JLabel` 时，可在 GUI 预览中直接编辑其中的文本。请输入 *`Your Name`*`'s Hi-Lo Guessing Game` 并按回车。如果以后要修改标签的文本，可在左下角的 Properties 面板中找到 `text` 属性，并输入所需的文本。当前，你可能只能在这个标签中看到文本的一部分，因为这个标签太小，无法显示完整的文本。请单击这个标签，其每个角上都将出现黑色的大小调整块。单击左下角并向左拖曳到内容面板的左边缘，再单击标签的右下角并拖曳到内容面板的右边缘。

现在，所有标签文本都出现在 GUI 预览顶部的标签中了。下面来将标签文本设置为粗体并居中。首先，在 Properties 面板中找到标签的 `horizontalAlignment` 属性，单击其值并从下拉列表中选择 CENTER。然后，找到标签的 `font` 属性，单击其值右边的三个句点，这将打开字体选择对话框，让你选择字体、样式和字号；我选择的是 Tahoma、15 点和 Bold，如图 3-11 所示。请注意，如果选择的字体较大，你可能需要调整标签的大小，以便能够容纳全部文本。

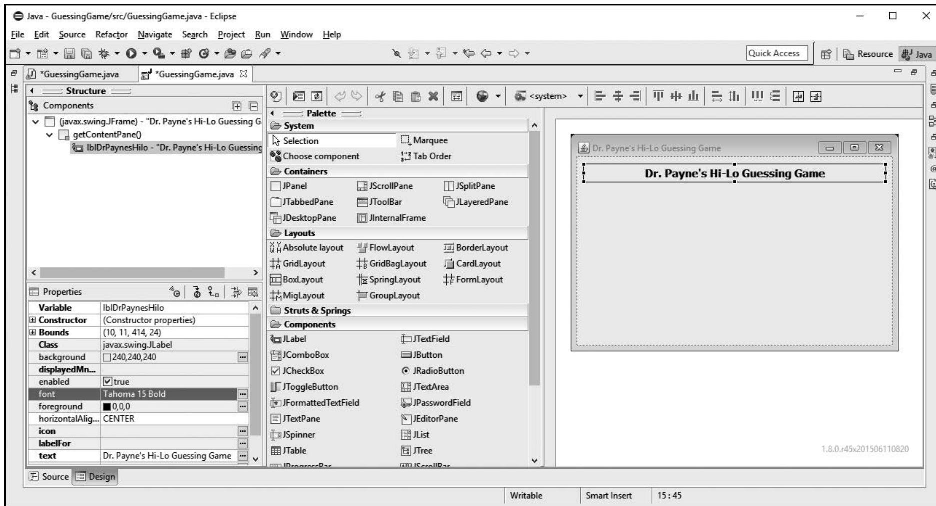


图 3-11　定制好的第一个标签，它位于内容面板顶部

下面再来添加一个标签，用于显示用户在这个游戏中看到的第一条提示语。单击 Palette 面板中的 JLabel，再将标签放在内容面板中央上方，并在其中输入文本 `Guess a number between 1 and 100`。你需要调整这个标签的尺寸，使其比文本宽些，以留下额外的边距。

下面在这个标签的右边放置一个文本框，供用户输入其猜测。为此，单击 Palette 面板中的 JTextField，并将这个文本框放在刚才添加的标签右边。

调整这个文本框的大小，使其足以容纳一个三位数字。单击刚添加的标签，并将其属性 `horizontalAlignment` 设置为 `RIGHT`，让文本离文本框更近些，如图 3-12 所示。

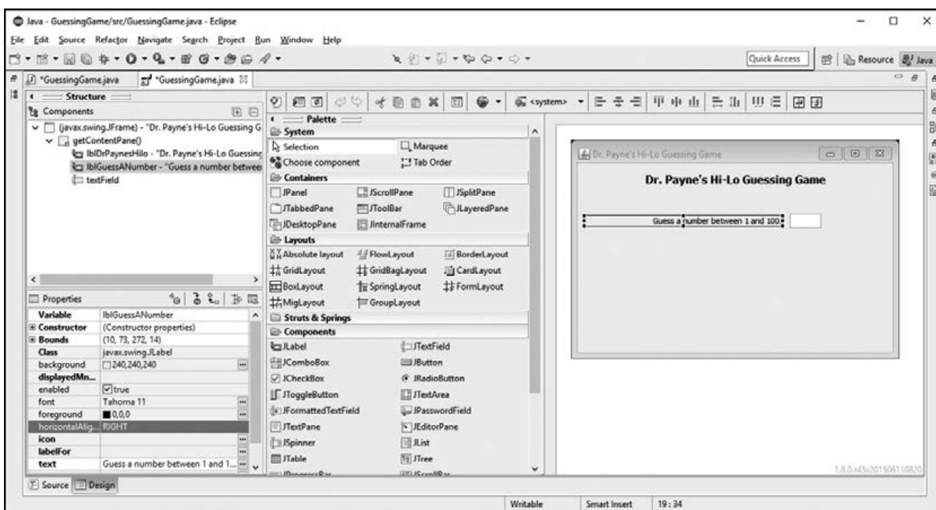


图 3-12　现在猜数游戏包含让用户进行猜测的标签和文本框

接下来，添加一个按钮，用户可通过单击它来提交猜测。在 Palette 面板中，找到并单击 JButton，将鼠标指向 GUI 预览窗口的中央并单击，将 JButton 放在这里，再将其文本改为 Guess!。

最后，在按钮下方添加一个 JLabel，将其文本设置为 Enter a number above and click Guess!，将其调整为与 JFrame 等宽，并将属性 horizontalAlignment 设置为 CENTER。在后面，你还将使用这个标签来告诉用户猜大了、猜小了还是猜对了。现在，GUI 布局类似于图 3-13，虽然组件之间没有很好地对齐。

添加所有的 GUI 组件后，便可做些微调，让布局居中并保持平衡，但在此之前，务必将文件存盘。

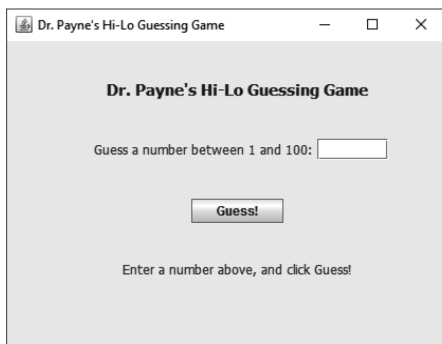


图 3-13　GUI 布局包含所有需要的组件，只是没有很好地对齐

3.4.3　对齐 GUI 元素

前面你都是通过目测来放置 GUI 组件，并尝试尽可能让间距正确并居中。但运行程序时，我们希望布局是完美的，所幸 Eclipse 内置了可帮助对齐组件的工具。

首先，我们让 4 个主要元素均匀地分布，即它们的垂直间隔相等。为此，请按住 Shift 键并单击，以同时选择 3 个标签和按钮，但不要选择供用户输入猜测的文本框。你将在 GUI 预览窗口上方看到一排小型对齐工具。请将鼠标指向每个工具，出现的工具提示指出了当前按钮是做什么的。如果不能看到所有的对齐工具，你可能需要调整 Eclipse 窗口的大小。

单击像一堆按钮的工具（图 3-14 所示最右边的图标），选定的 4 个元素将在垂直方向上均匀地分布。



图 3-14　使用 GUI 预览窗口上方的工具对齐组件并使其均匀分布

现在只选择文本框，并移动它，使其重新与提示用户猜数的标签对齐。前面之所以没有选择文本框，是因为如果这样做，Eclipse 将均匀地分布全部 5 个组件，将文本框及其配套标签分开，从而排成 5 行而不是 4 行。

注意 在 WindowBuilder 对齐组件或执行其他操作时，如果结果一团糟，可按 CTRL-Z（或⌘-Z）来撤销最后一个操作。这是 WindowBuilder Editor 提供的一项极佳的安全保障功能，让你能够大胆地尝试新鲜事物，而不用担心布局变得一团糟。

最后，单击按钮 Guess!，再单击 GUI 预览上方的对齐按钮（右数第二个按钮，其工具提示为“Center horizontally in window”）。如果你愿意，也可让其他组件居中或进行其他调整。如果你要同时移动多个组件，可按住 Shift 键并单击以同时选择它们，再通过单击并拖曳来移动它们。

对布局满意后，该给组件命名，为编写代码做好准备了，但在此之前，请保存所做的修改。

3.4.4 给 GUI 组件命名以方便编写代码

对玩家来说，猜数游戏的用户界面已经就绪，但为方便编程，还需在 Java 源代码文件中做些调整（后续所有编程工作都将在这个文件中进行）。我们需要给每个组件命名，以便在源代码中引用它们。

这一步有时被称为“关联”，因为我们将把每个 GUI 组件都与变量相关联，以便能够在 Java 源代码中访问它们。每当我们添加 GUI 元素时，Eclipse 都会给它命名，但我们想修改这些名称。请选择供用户输入猜测的文本框，此时如果你查看 Properties 面板，将发现其中的第一个属性为 `Variable`。对于你添加的第一个文本框，Eclipse 指定的默认变量名可能是 `textField`。请单击 `Variable` 属性旁边的值，并将表示该文本框的变量重命名为 `txtGuess`，如图 3-15 所示。

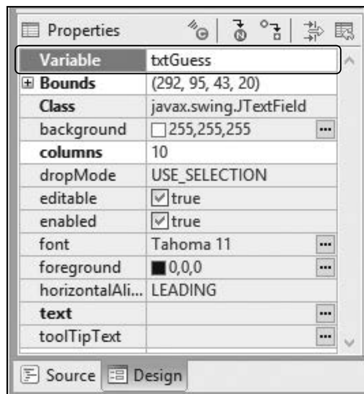


图 3-15 将表示每个 GUI 元素的变量重命名，以便能够在 Java 源代码中轻松地引用 GUI 元素

看到名称 `txtGuess`，我们就知道它是一个文本框，存储的是用户的猜测。你将发现，通过以一致而符合逻辑的方式给 GUI 组件命名，可提高编程的速度和效率，还可减少错误。

给文本框重命名后，选择 GUI 窗口底部的标签，其文本为“Enter a number above and click Guess!”。这个标签还将用于向用户显示信息，如 `Too high`、`Too low` 或 `You win!`，因此我们将其

重命名为 lblOutput。看到这个名称，我们就知道它是一个 GUI 标签（lbl），用于显示要供用户查看的输出。

现在，如果你单击 WindowBuilder Editor 左下角的 Source 选项卡，将看到 Eclipse 编写了相应的 Java 源代码，以生成你设计的 GUI 布局，如图 3-16 所示。

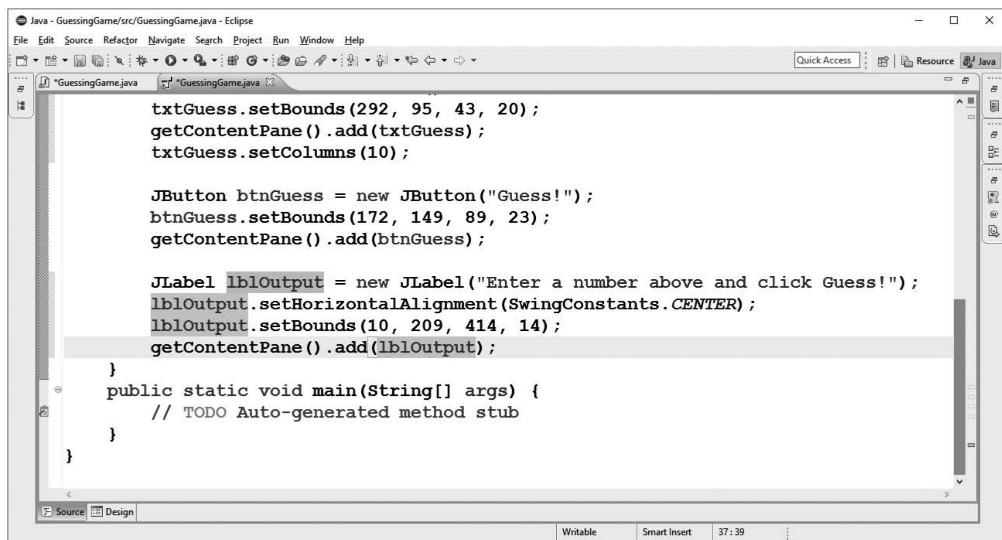


图 3-16　单击选项卡 Source，你将看到 Eclipse 为我们编写了生成 GUI 组件的 Java 代码

请注意，在源代码中使用了你给文本框和输出标签指定的变量名（txtGuess 和 lblOutput）。现在请保存前面所做的所有修改。

然而，着手给猜数游戏编写代码前，先来调整一下源代码。

3.4.5　将 GUI 与 Java 代码相关联

向上滚动到 GuessingGame 类开头，你将发现 Eclipse 在这里声明了 JTextField txtGuess：

```
public class GuessingGame extends JFrame {
    private JTextField txtGuess;
```

对于文本框，Eclipse 默认这样做是有原因的。通过在类级声明文本框，使得在类的任何地方都可访问它（例如，检查用户猜大了还是猜小了）。文本框通常都是这样使用的，因此 Eclipse 在 GuessingGame 类的开头声明私有的 JTextField txtGuess，而不是在这个类的方法（函数）中声明它。在面向对象编程中，使用限定符 private 是一种**最佳实践**（推荐做法）。声明为私有的可让其他类看不到它。由于 txtGuess 是私有的，除 GuessingGame 外的其他类将无法访问（或不小心修改）它，这正是我们想要的。

在这个猜数游戏中，我们希望能够访问用户在文本框 txtGuess 中输入的值，因此 Eclipse 在

类级声明了它是件大好事。我们还要访问标签 lblOutput，以便能够修改向用户显示的文本，指出他猜大了、猜小了还是猜对了。因此，我们需要在 GuessingGame 类的开头（紧跟在 txtGuess 声明的后面）再添加一个声明：

```
public class GuessingGame extends JFrame {
    private JTextField txtGuess;
    private JLabel lblOutput;
```

这行代码将 lblOutput 声明为变量，它将指向这个应用程序的 GUI 布局中的一个 JLabel 对象。我们将这个对象引用声明为私有的，以对外隐藏这项数据，但在整个 GuessingGame 类中都可使用它。

需要做的最后一项修改是校正原来声明 lblOutput 的代码行，它位于 GuessingGame 类的末尾附近。请向下滚动，找到类似于下面的代码行：

```
    JLabel lblOutput = new JLabel("Enter a number above and click Guess!");
```

再将其修改成下面这样：

```
    lblOutput = new JLabel("Enter a number above and click Guess!");
```

注意，我们删除了多余的 JLabel 声明。如果不这样做，程序将不能正确工作，因为你已经在类开头将 lblOutput 的类型声明为 JLabel。如果你留下第二个 JLabel 声明，Java 将认为你要创建两个名称都为 lblOutput 的 JLabel 对象。通过删除第二个 JLabel 声明，你告诉 Java 要将程序开头创建的 JLabel lblOutput 的文本值初始化为 Enter a number above and click Guess!。

既然我们都在 GuessingGame 类开头声明变量了，就顺便再添加一个重要的变量（theNumber）吧。别忘了，在基于文本的猜数游戏中，这个变量用于存储要让用户去猜的随机数。请在 txtGuess 和 lblOutput 后面添加 theNumber 的声明，如下所示：

```
public class GuessingGame extends JFrame {
    private JTextField txtGuess;
    private JLabel lblOutput;
    private int theNumber;
```

在类开头正确地声明变量后，就可开始给 GUI 版猜数游戏编写代码了。下一节将介绍如何获取用户在 GUI 文本框中输入的猜测，并检查它是太大还是太小了；我们还将在 GUI 标签 lblOutput 中显示输出文本，帮助用户决定接下来如何猜。

3.5　添加检查用户猜测的方法

通过添加私有变量 txtGuess 和 lblOutput，将 GUI 与 Java 源代码相关联后，接下来将着手实现猜数游戏的逻辑。

我们在 GuessingGame 类开头附近编写方法 checkGuess()，其签名（骨架）类似于下面这样：

```
public class GuessingGame extends JFrame {
    private JTextField txtGuess;
    private JLabel lblOutput;
    private int theNumber;
    public void checkGuess() {
    }
```

请注意，这个方法被声明为公有的。在类中，变量通常声明为私有的，但操作这些变量的方法或函数通常声明为公有的，以便其他类能够调用它们。可将类视为支票账户，其余额是私有的，只有你和银行能够访问，但存款函数是公有的，这样别人才能将钱存入你的账户。

第二个术语（void）告诉 Java，我们不要求这个函数**返回**值（返回值是函数或方法的输出）。例如，将华氏温度转换为摄氏温度的函数可能接受一个表示华氏温度的值，并返回一个表示转换得到的摄氏温度的数字值。在这个猜数游戏中，方法 checkGuess()不会返回类似于这样的信息。相反，我们将让它通过图形用户界面向用户显示信息，因此我们将返回类型设置为 void，这表示不返回值。

最后，我们将这个方法命名为 checkGuess()，并提供左大括号和右大括号，它们将包含告诉程序如何对用户猜测进行检查的代码。

3.5.1 获取 JTextField 中的文本

下面来给方法 checkGuess()添加代码，使其获取用户输入的字符串，并将其与要猜的数字进行比较。请在方法 checkGuess()的两个大括号之间添加如下代码行：

```
    public void checkGuess() {
        String guessText = txtGuess.getText();
    }
```

这行代码新建一个名为 guessText 的 String 变量，用于存储用户在 GUI 文本框中输入的数字。为获取用户输入的文本，我们需要调用方法 txtGuess.getText()；然后，将结果存储在新创建的字符串变量 guessText 中。

当你在 txtGuess 后面加上句点（.）时，可能出现一个弹出窗口，如图 3-17 所示。

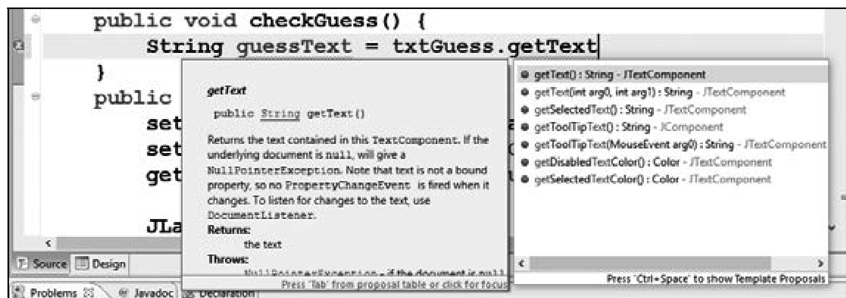


图 3-17 在 txtGuess 后面添加句点运算符（.）后，可能出现很有帮助的代码推荐窗口

这被称为**内容助手**（content assist）——Eclipse 试图通过提供补全当前语句的代码来施以援手。这种内容助手也被称为**代码推荐器**（code recommender），因为它推荐常见的代码选项，如当前类或对象的方法。Eclipse 之所以被专业 Java 开发人员视为强大的 IDE，它提供的代码推荐功能是原因之一。在大多数情况下，选择代码推荐器的建议更快捷，还有助于避免错误，这可提高程序员开发复杂应用程序的效率。

接下来，需要再创建一个字符串变量，用于存储指出用户猜大了、猜小了还是猜对了的消息。下面就来声明这个变量：

```
public void checkGuess() {
    String guessText = txtGuess.getText();
    String message = "";
}
```

我们还不知道用户是猜大了还是猜小了，因此使用一对双引号（""）将 message 初始化为空字符串。

将用户的猜测与随机数（theNumber）进行比较后，我们将更新这个字符串，再将其输出到 GUI 标签中。要检查用户的猜测，需要将他输入的文本（当前存储在 guessText 中）转换为数字，这样才能与 theNumber 进行比较。

3.5.2　将字符串转换为数字

在第 2 章创建的这个应用程序的基于文本的版本中，我们使用了一个 Scanner 对象来从用户通过键盘输入的文本中获取数字。在 GUI 版本中也可这样做，但有一种更紧凑的方式，而且效果更好。

为了让我们能够生成随机数，Math 类提供了方法 Math.random()。同样，有一个用于处理整数的类，它就是 Integer。Integer 类提供了多个可帮助处理整数的函数，其中的 Integer.parseInt()用于在文本字符串中寻找整数。请在代码行 String message = "";后面输入如下代码行。

```
public void checkGuess() {
    String guessText = txtGuess.getText();
    String message = "";
    int guess = Integer.parseInt(guessText);
}
```

这行代码首先声明了整型变量 guess。接下来，它在用户输入的文本中搜索（**解析**）出一个整数。例如，字符串"50"将变成**数字** 50。最后，它将这个数字存储到变量 guess 中。我们需要的是用户猜测的数字版本，这样才能使用比较运算符<和>与 theNumber 进行比较。

将用户的猜测存储到变量 guess 中后，就可将其与计算机生成的随机数 theNumber 进行比较了。我们还未编写将随机数存储到 theNumber 中的代码，但马上就会这样做。比较 guess 和 theNumber 的方式与基于文本的版本中类似，但不使用 System.out.println()打印到控制台，而是将输出存储在前面创建的字符串变量 message 中：

```
        String guessText = txtGuess.getText();
        String message = "";
        int guess = Integer.parseInt(guessText);
        if (guess < theNumber)
            message = guess + " is too low. Try again.";
        else if (guess > theNumber)
            message = guess + " is too high. Try again.";
        else
            message = guess + " is correct. You win!";
    }
```


请注意，这 3 条 if-else 语句几乎与基于文本的版本中相同，只是我们将输出消息（太大、太小或正确）存储在一个变量中，而不是直接将它输出到控制台窗口。依然需要向用户显示 message，但我们将使用 GUI 标签 lblOutput 来完成这项任务。为此，我们使用方法 setText()，如下所示：

```
        else
            message = guess + " is correct. You win!";
        lblOutput.setText(message);
     }
```

这条语句将标签 lblOutput 的 text 属性设置为 String 变量 message 的值（随用户的猜测而异），从而在 GUI 窗口中向用户显示输出消息。

方法 checkGuess()最终的代码应类似于下面这样：

```
    public void checkGuess() {
        String guessText = txtGuess.getText();
        String message = "";
        int guess = Integer.parseInt(guessText);
        if (guess < theNumber)
            message = guess + " is too low. Try again.";
        else if (guess > theNumber)
            message = guess + " is too high. Try again.";
        else
            message = guess + " is correct. You win!";
        lblOutput.setText(message);
    }
```

接下来，需要编写在 theNumber 中存储一个随机数的代码。为此，将使用方法 newGame()，因为我们要在用户每次开始新游戏时都生成一个随机数。

3.6 开始新游戏

我们要让计算机在每次开始新一轮游戏时都生成一个随机数，因此在 GuessingGame 类的一个方法中完成这项任务比较合理，这样就可在用户获胜并要接着玩时调用这个方法。

方法 newGame()的签名与方法 checkGuess()类似。请将它放在 checkGuess()的右大括号和 public GuessingGame()之间：

```
        lblOutput.setText(message);
    }
    public void newGame() {
    }
    public GuessingGame() {
```

我们将这个方法声明为公有的，这样可从外部类中调用它。我们还需让 Java 知道，这个方法与 checkGuess()一样，也不需要返回信息，因此其返回类型为 void。最后，这个方法名为 newGame，它后面是一对括号（()）。当前，这个方法的方法体是空的（大括号之间什么都没有），但我们马上就会解决这个问题。

开始新游戏意味着让计算机生成一个随机数。将随机数赋给变量 theNumber 的代码与基于文本的版本中类似，只是这里将这些代码放在一个方法中。因此，最终的方法 newGame()类似于下面这样：

```
    public void newGame() {
        theNumber = (int)(Math.random() * 100 + 1);
    }
```

就这么简单！现在，所需代码都已就绪，只需将它们与 GUI 关联起来，就可创建出管用的猜数游戏。只要能够监听用户事件，从而在用户单击 Guess!按钮时做出响应，这个游戏就可以玩了！

3.7　监听用户事件——单击 Guess!按钮

要运行这个应用程序，需要完成的最后一步是将按钮 btnGuess（包含文本“Guess!”的按钮）同检查用户猜测的方法（checkGuess()）关联起来。在 Java 中，可使用**事件监听器**来完成这项任务。事件监听器是让程序等待（**监听**）用户交互**事件**（如按钮单击、鼠标移动或键盘输入）的代码。

请单击选项卡 Design 切换到 GUI 预览。在 GUI 预览窗口中，找到 Guess!按钮（如图 3-18 所示），再双击它。

双击 Guess!按钮将自动切换到 Source 选项卡，你将发现 Eclipse 在其中添加了新代码。这些新代码是一个事件监听器的骨架,这里是用户单击按钮 btnGuess 时发生的事件的 ActionListener()。

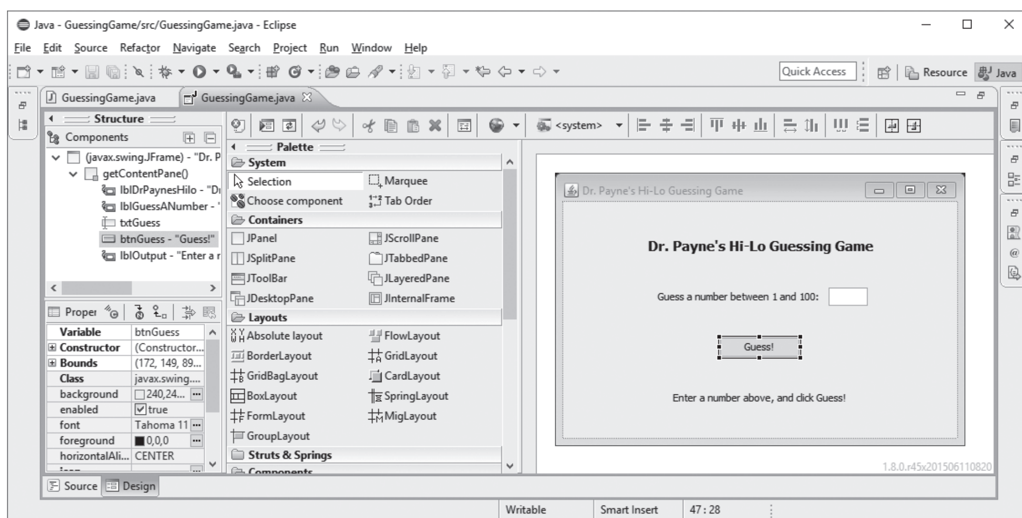


图 3-18　在 Design 视图中，双击 Guess!按钮在代码中添加一个事件监听器

除了在用户单击这个按钮时我们要采取的措施外，其他的事情 Eclipse 都替我们完成了。我们来看看 Eclipse 创建的事件监听器代码。

```
  JButton btnGuess = new JButton("Guess!");
❶ btnGuess.addActionListener(new ❷ActionListener() {
    ❸public void actionPerformed(ActionEvent e) {
      }
  });
  btnGuess.setBounds(172, 149, 89, 23);
  getContentPane().add(btnGuess);
```

在❶处，使用方法 addActionListener()将处理按钮单击的事件监听器 ActionListener()关联到了 btnGuess。接下来的两行代码位于方法 addActionListener()的括号内。

在方法 addActionListener()的调用中，我们首先看到的是关键字 new，这表明 Eclipse 创建了一个新的 ActionListener 对象❷。不同于我们在本书前面看到的对象，这个新对象是一个**匿名内部类**，这意味着没有变量指向它，而且它是在 GuessingGame 类中创建的。匿名内部类可帮助提高编程速度，因为无需创建独立的类来处理简短、快捷的事件，如按钮单击；相反，我们可在创建按钮的地方插入事件处理程序的代码。在这里，ActionListener 只监听按钮单击事件，因此非常适合使用匿名内部类。

Eclipse 创建了匿名内部类的骨架，但我们还需要编写指定按钮行为的代码。这些代码将放在方法 actionPerformed()的大括号内❸。我们需要在方法 actionPerformed()中指定程序在用户单击 btnGuess 按钮时该如何做。这个方法将一个 ActionEvent 作为参数，这个参数表示监听器监听的事件（用户操作）。在这里，这个 ActionEvent 参数被命名为 e，但你可使用任何变量名。用户输入猜测并单击 Guess!按钮时，这个操作将被赋给 e，而我们需要使用前面创建的方法 checkGuess()

来检查用户的猜测：

```
btnGuess.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        checkGuess();
    }
});
```

多亏了 Eclipse，我们只需添加语句 checkGuess();。自动代码完成功能替我们节省了时间，还有助于避免输入错误。

添加这些代码后，现在只需告诉应用程序在启动时如何做即可。这样做之前，请保存前面所做的工作。

3.8　设置 GUI 窗口

要运行这个游戏，还需告诉它如何设置 GUI 窗口。这包括创建 GUI、调用方法 newGame()来开始新游戏，以及设置窗口的宽度和高度。

请找到文件末尾附近的 main()方法，并在其大括号内添加如下代码行：

```
    public static void main(String[] args) {
     ❶ GuessingGame theGame = new GuessingGame();
     ❷ theGame.newGame();
    }
}
```

❶处的代码创建一个名为 theGame 的 GuessingGame 对象；❷处的代码启动猜数游戏——生成一个随机数。

下面来告诉 Java 我们希望窗口多大。对于 GUI 桌面应用程序，要指定其窗口的尺寸，可调用方法 setSize()并传入高度和宽度值。

```
    public static void main(String[] args) {
        GuessingGame theGame = new GuessingGame();
        theGame.newGame();
        theGame.setSize(new Dimension(450,300));
```

方法 setSize()接受一个类型为 Dimension 的参数，这是 java.awt 库［awt 是 abstract window toolkit（**抽象窗口工具包**）的缩写］的一个内置类型。在上述调用 setSize()的代码中，括号内的 new Dimension()让 Java 创建一个高 450 像素、高 300 像素的 Dimension 对象。但我们还没有导入 java.awt 库，这意味着 Eclipse 不知道 Dimension 为何物，因此 Eclipse 编辑器将在类名 Dimension 下方加上红色线条，指出这条语句存在错误。如果你单击编辑器中的 Dimension，Eclipse 将显示一个快速修复（Quick Fix）列表，如图 3-19 所示（如果单击不能显示快速修复列表，请将光标放在 Dimension 内，再按 CTRL-1 或⌘-1）。

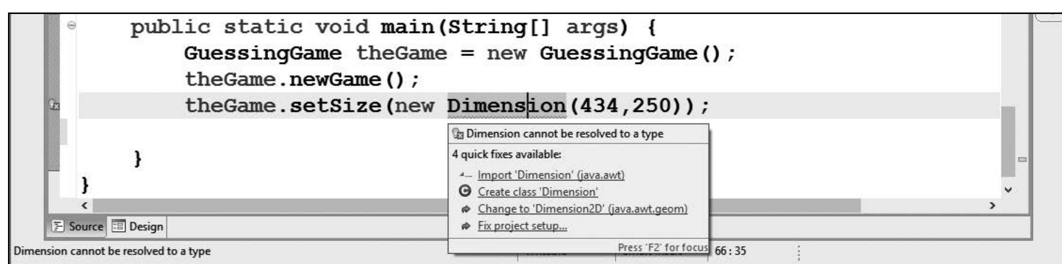


图 3-19　对于我们输入的新代码，Eclipse 推荐的快速修复选项；请选择第一个选项以导入相应的类

我们要使用 java.awt 包中的 Dimension 类，因此第一个快速修复选项 Import 'Dimension' (java.awt)是正确的选择。请单击该选项，Eclipse 将在文件开头的导入列表中添加 java.awt.Dimension。

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.SwingConstants;
import java.awt.Dimension;
import java.awt.Font;
import javax.swing.JTextField;
```

最后，在方法 main()中添加如下代码行，让应用程序出现在屏幕上：

```
        theGame.setSize(new Dimension(450,300));
        theGame.setVisible(true);
    }
```

这行代码对 GuessingGame 对象 theGame 调用方法 setVisible()。别忘了，GuessingGame 扩展了 JFrame 类，因此 theGame 也是父类 JFrame 的**后代**。我们要将包含 GUI 版猜数游戏的 JFrame 的 visible 设置为布尔值 true。

布尔值只有两个——true 和 false（它们都是全小写），它们用于布尔表达式中。**布尔表达式**是结果为 true 或 false 的表达式。方法 checkGuess()使用的 if 语句中的表达式，如(guess < theNumber)就是布尔表达式。当我们计算这个表达式——检查 guess 是否小于 theNumber 时，结果要么为 true（意味着 guess 小于 theNumber），要么为 false（guess 不小于 theNumber）。这个布尔表达式的结果决定了是否会执行相应 if 语句后面的语句。

GuessingGame 类的方法 main()的完整代码如下：

```
    public static void main(String[] args) {
        GuessingGame theGame = new GuessingGame();
        theGame.newGame();
        theGame.setSize(new Dimension(450,300));
        theGame.setVisible(true);
    }
```

3.9　开玩

代码编写完成后，请将文件 GuessingGame.java 存盘，再单击 Run 按钮或选择菜单 Run▶Run。游戏窗口将显示出来。

尝试输入一些数字，看看应用程序是否管用。在玩这个游戏的同时，别忘了检查每个组件，看看它们是否管用。务必测试按钮 Guess!、底部的标签以及文本框。如果发现问题，回过头去检查代码。请注意错误在什么时候或什么地方发生，以缩小查找范围。随着你不断地编写代码，你查找错误的能力将越来越强。图 3-20 显示了这个应用程序能够正确运行时是什么样的。

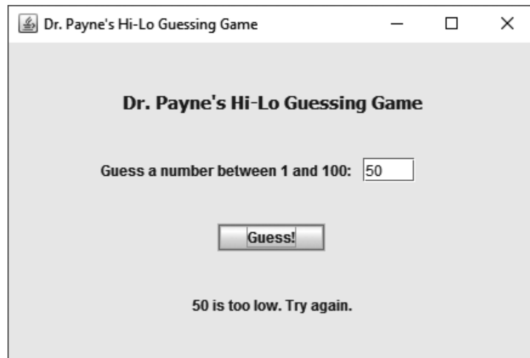


图 3-20　按钮 Guess!管用!

确定游戏在各方面都没问题后，就可进入下一步了。

当前，这个游戏没问题，但你获胜后就结束了，因为它没有包含在用户获胜后让 Java 开始新游戏的代码。

下面将首先设置这个游戏，使得用户获胜后还能重玩。然后，对用户界面做些细微的调整，以改善用户体验。

3.10　添加重玩功能

与基于文本的猜数游戏一样，我们来添加重玩功能。前面创建了方法 `newGame()`，但仅在启动游戏时在方法 `main()`中调用了它。

我们需要修改 GUI 或改变应用程序的行为。可添加一个 Play Again 按钮，用户可在要再玩一轮时单击它，也可让应用程序在用户获胜后自动开始新游戏。

添加 Play Again 按钮的优点在于，从用户的角度看很容易理解。然而，我们不希望按钮 Play Again 始终出现在 GUI 窗口中，因为仅当用户获胜后才需要它。一种办法是添加一个按钮，但在游戏运行时让它不可见，等到用户获胜后再让它可见。出于简化考虑，这里不会这样做，但如果你想尝试这样做，请参阅本章末尾的编程练习 2。

这里不添加按钮，而修改应用程序的行为，使其在用户获胜后自动生成随机数。为此，只需

修改程序的一个地方：处理用户的猜测与随机数相同的 else 语句。用户赢得一轮后，应自动开始新的一轮。这简单明了：不需要额外的按钮，也无需用户做决定。如果用户不想玩了，只需关闭窗口即可。我们将通过修改如下代码行来添加这种行为：

```
else
    message = guess + " is correct. You win!";
```

我们将调用 newGame()来开始新游戏。为在这条 else 语句中调用方法 newGame()，需要将两条语句（设置变量 message 的语句和调用 newGame()的语句）组合成一个代码块，如下所示：


```
else {
    message = guess +
            " is correct. You win! Let's play again!";
    newGame();
}
```

大括号将这两条语句组合成一个代码块，每当这条 else 语句被选作正确的程序路径时，都将执行这个代码块。如果没有大括号，将只执行 else 语句后面的第一条语句，而方法 newGame()将在 if-else 语句执行完毕后被调用，导致程序在用户每次猜测后都改变秘密数字。

另外请注意，我们在消息字符串末尾添加了 Let's play again!，让用户知道要继续玩只需接着猜就可，而计算机在每轮都会重新选择随机数。完成这些修改后，将程序存盘，再运行它看看这个新版游戏！

3.11 改善用户体验

在玩这个游戏的过程中，你可能发现添加一些游戏元素可改进用户界面。例如，如果这个程序能够在用户单击 Guess!按钮或按回车时接受猜测，**用户体验**（UX）将更自然、更直观。

另一个问题是，为再次猜测，用户需要选择原来的数字并输入新数字（或按 DELETE 或 BACKSPACE 键）来清除前一次的猜测。另外，用户还必须单击文本框，因为焦点已切换到 Guess!按钮。为改进用户界面，我们可让光标在用户提交猜测后回到文本框，并自动选择或删除前一次猜测的数字，让新猜测替换旧猜测。

下面就来解决这两个问题。

3.11.1 让用户能够按回车键来提交猜测

我们要做的第一项改进是，让用户能够按回车键来提交猜测。为此，切换到 Design 选项卡，再右击 Guess!按钮，并在弹出菜单中选择 Add event handler▸action▸actionPerformed。

就像前面为 btnGuess 创建事件处理程序时一样，Eclipse 将自动添加为 txtGuess 创建操作监听器的代码，如下面的代码片段所示。另外，就像前面所做的一样，我们将在这些代码的 actionPerformed()部分的大括号内调用方法 checkGuess()。对于文本框，我们通常只处理一个操

作事件，那就是用户按回车键。

```
txtGuess.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        checkGuess();
    }
});
```

添加这行代码后，将文件存盘，并再次运行程序。现在，当你按回车键时，将调用方法 checkGuess()。Guess!按钮依然管用，因此用户可选择使用键盘或鼠标与程序交互。

3.11.2　自动删除前一次猜测的数字

现在来解决必须单击文本框并将原来的猜测删除的问题。为此，需要学习 JTextField 对象（如 txtGuess）的另外两个方法。

首先，必须考虑要在什么时候选择文本。就这个猜数游戏而言，让用户能够再次猜测的恰当时机是检查用户当前的猜测后。在代码中，这是方法 checkGuess()的末尾，因为我们要在用户猜测并准备再次猜测后选择文本。

因此，我们需要在方法 checkGuess()末尾编写代码，将光标放回到文本框 txtGuess 并选择用户前一次猜测的数字。但将光标放在文本框中以及选择文本框中所有文本的方法是什么呢？Eclipse 的内容助手功能正好能够派上用场。输入对象名 txtGuess 和句点运算符（.），Eclipse 将显示一个列表，其中包含所有可用的方法，如图 3-21 所示。

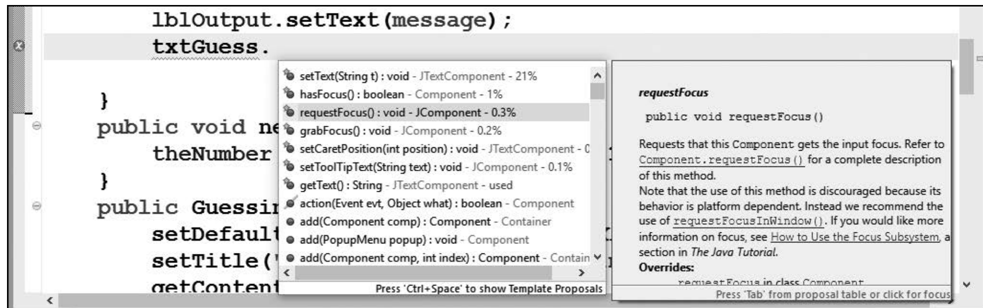


图 3-21　内容助手功能显示可通过 JTextField txtGuess 调用的所有方法

如你所见，可通过 JTextField 对象（如 txtGuess）调用的方法有数十个，但我们需要的是 requestFocus()。这个方法的文档指出："请求让当前组件获得输入焦点。"对文本框而言，**焦点**就是光标。这个方法的功能正是我们需要的：每当用户猜测后，都将光标放回到文本框中。

最后，我们要选择用户前一次猜测时输入的所有文本，让它们自动被用户的新猜测替换。要选择文本框中的所有文本，可使用方法 selectAll()，它也包含在 Eclipse 内容助手功能的建议中。请在方法 checkGuess()的末尾（代码行 lblOutput.setText(message);和右大括号之间）添加如下两行代码：

```
        lblOutput.setText(message);
        txtGuess.requestFocus();
        txtGuess.selectAll();
    }
```

再将应用程序存盘。现在如果你再次运行这个游戏，将感觉玩起来流畅、直观得多。这个猜数游戏的 GUI 优雅而易于使用，提供了流畅的用户体验。这个游戏易学难精，就像我们玩的其他游戏一样，但它是我们使用 Java 从头创建的，同时由于使用了 Swing 工具包，其外观显得非常专业。

当前，这个应用程序在各方面都很出色，只有一个方面例外，就是安全。你可能会说："什么？安全？我还以为我创建的是一个有趣的 GUI 游戏呢！"确实如此。这个程序做得很好，但仅在用户正确地输入了数字时才如此。

如果用户手滑，在没有输入猜测时就按下回车键或单击 Guess!按钮，结果将如何呢？如果用户输入的是非整数文本（如 `blah`），结果又将如何呢？图 3-22 显示了用户输入无效时 Eclipse 中出现的情况。

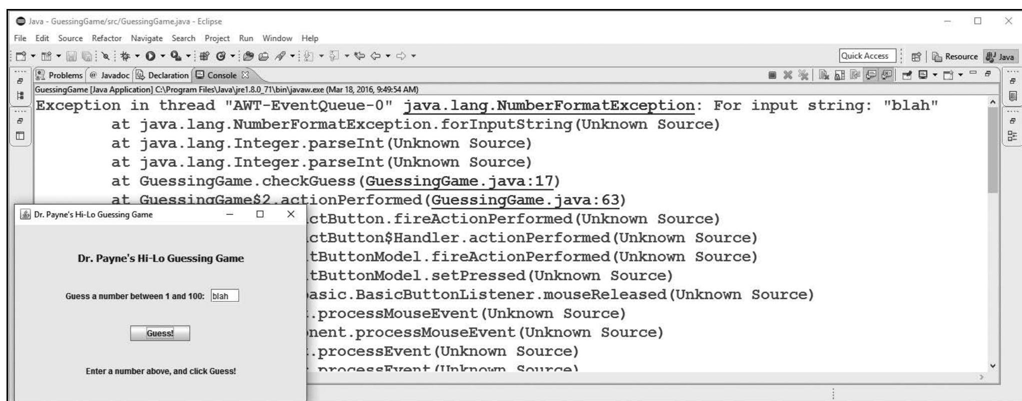


图 3-22 用户在文本框中输入无效数据（如 `blah`）带来的恶果

如你所见，控制台窗口充斥着错误。无效输入引发了**异常**——意外或导致故障的情景。图 3-22 中的第一行输出指出，出现了异常 `NumberFormatException`，因为字符串`"blah"`无法转换为数字。你需要学习如何在程序中处理异常。

3.12 处理无效的用户输入

程序运行时可能出现的糟糕情况称为异常，而 Java 提供了预测并处理异常的可靠途径。除前面讨论的无效用户输入外，异常还包括文件缺失（文件被删除或意外地拔出了 USB 驱动器）和网络问题（如连接缺失或服务器故障）。

编写用于处理异常的代码被称为**异常处理程序**。异常处理程序用于应对异常情况，但我们怎

么知道在什么情况下可能发生异常呢？我们必须预测程序运行时可能发生的所有糟糕情况吗？

很多可能出错的情形都是特定类型的事件，如用户输入、文件输入/输出或网络接入。就拿刚才列出的异常来说吧，每当我们让用户提供特定类型的输入时，都可能出现输入无效的情况；每当我们读写文件，如在照片应用程序中查看和保存图片时，都可能出现文件缺失的问题。如果我们能够预测可能发生的异常，进而编写异常处理程序来应对这些异常，代码将更安全、更可靠，抵御恶意攻击者通过输入无效输入来破坏程序的能力也更强。

就这里的猜数游戏而言，当我们让用户输入一个数字，而用户却输入文本时，将发生异常。具体地说，当我们试图从不包含整数的字符串获取整数时，将触发异常。有意或无意的无效用户输入，是导致异常的常见罪魁祸首，例如，攻击者试图有意提供无效输入来破坏你的程序。

Java 通过 try-catch 语句提供了整洁的异常处理方式。try-catch 语句的基本思想是，让程序尝试执行可能引发异常的操作，并在异常发生时进行**捕获**（处理），以免它破坏程序。catch 块仅在发生异常时才会执行。

try-catch 语句的最后一部分是可选的 finally 块。无论是否发生异常，try-catch-finally 语句中的 finally 块都会执行。如果没有发生异常，finally 块将在 try 块执行完毕后执行；如果发生异常，finally 块将在执行完 catch 块后执行。

try-catch 语句的格式通常类似于下面这样（这只是一个说明格式的示例，请不要将其输入到你的代码中）：

```
try {
    // 可能引发异常的操作，如获取的输入无效
} catch (Exception e) {
    // 处理异常或从异常中恢复，如让用户提供有效输入
} finally {
    // 收尾代码；无论是否发生异常都会执行
}
```

注意，try-catch-finally 语句的每部分的代码都放在一对大括号内，就像 if-else 和循环语句一样。在 catch 部分，必须指定一个 Exception 参数，这可以是具体的异常，如 NumberFormat Exception，也可以是基类 Exception。这个参数将获取有关异常的信息，如异常发生在哪一行。

要在这个游戏的代码中正确地添加 try-catch-finally 块，需要考虑哪些地方可能发生异常（进而将相关的代码放在 try 块中）、发生异常时要采取什么措施（进而将相关的代码放在 catch 块中），以及执行完 try 或 catch 块要如何做（进而将相关的代码放在 finally 块中）。

我们尝试从用户输入的文本中提取整数时，可能出现输入无效问题，因此需要将这些代码放在 try 块中。我们在从文本中提取整数的代码行前面，添加关键字 try 和左大括号：

```
public void checkGuess() {
    String guessText = txtGuess.getText();
    String message = "";
    try {
        int guess = Integer.parseInt(guessText);
        if (guess < theNumber)
```

如果用户输入的不是数字，就无法判断他猜大了还是猜小了，因此相应的 if-else 语句也应包含在 try 块内。请在相应的 else 语句的右大括号后面，加上 try 块的右大括号，如下面的代码清单所示。接下来需要添加一个 catch 块。在这个游戏中，如果用户输入无效，我们要让他输入 1~100 的整数。为此，可使用后面将显示在标签 lblOutput 中的字符串变量 message，因此 catch 块应类似于下面这样：

```
        else {
            message = guess +
                    " is correct. You win! Let's play again!";
            newGame();
        }
    } catch (Exception e) {
        message = "Enter a whole number between 1 and 100.";
    }
```


无论是否发生异常，我们最后要做的都是显示输出消息，并为用户再次输入做好准备：让文本框获得焦点并选择文本框中的所有文本。这 3 条语句将放在 finally 块中，而 finally 块位于刚才添加的 catch 语句的右大括号后面：

```
        } finally {
            lblOutput.setText(message);
            txtGuess.requestFocus();
            txtGuess.selectAll();
        }
    }
    public void newGame() {
```

你可能注意到了，缩进了 try、catch 和 finally 块中的语句，让代码更容易阅读。别忘了，Eclipse 提供了一种快捷方式，修改代码后可使用它来自动确保缩进正确。为此，只需选择所有的代码，再选择菜单 Source▶Correct Indentation。

保存应用程序，并运行几次以测试用户界面。尝试多次输入非整数，以确定异常处理像要求的那样工作。如果遇到导致程序无法编译的错误，请检查大括号是否匹配。每次将代码组合成代码块后，如果发生错误，都应首先检查大括号是否匹配。添加 try-catch-finally 语句后，GUI 版猜数游戏将更可靠、更安全、更可预测。

安全编程绝非装饰，而是优秀软件至关重要的组成部分。就这里的猜数游戏而言，代码不可靠的后果好像没什么大不了，但请想想运行在医疗设备、卫星、汽车、飞机或电网中的代码吧，可靠的代码对我们的安全和保障至关重要。即便在智能手机上运行小型游戏或应用程序，如果其代码不可靠，也可能将你的数据暴露给黑客。在任何时候，安全编程都是明智的做法。

3.13　小结

通过开发 GUI 版猜数游戏，你学到了如下 Java 编程技能：

- 导入 javax.swing 包以创建 GUI 窗口；
- 在 Eclipse 中创建 GUI 应用程序；
- 使用 WindowBuilder Editor 设计 GUI；
- 使用 Properties 面板修改 GUI 组件的属性；
- 以一致的方式给 GUI 组件命名，以方便在 Java 代码中使用它们；
- 将 GUI 元素关联到 Java 变量和代码；
- 获取用户在 JTextField 中输入的文本并在程序中使用它们；
- 监听用户事件，如按钮单击；
- 从用户的角度测试应用程序，以改善用户体验；
- 使用 try-catch-finally 处理异常。

3.14　编程练习

为复习并使用学到的知识，以及获得更多的编程技能，请尝试完成这里的编程练习。如果遇到困难，可从 https://www.nostarch.com/learnjava/下载示例解决方案。

3.14.1　编程练习 1：告诉用户他猜了多少次

请尝试修改获胜消息，在用户获胜时指出猜了多少次才猜对，如下所示：

```
62 is correct! You win after 7 tries!
```

为完成这项任务，你需要创建一个新变量（如 numberOfTries）来存储用户猜测的次数，在每次调用方法 checkGuess()时都将猜测次数加 1，并在用户获胜时显示猜测次数。

你也可以在游戏开始时将猜测次数设置为 7 或 8，再在用户每次猜测后都将猜测次数减 1。猜测次数变为零时，告诉用户他输了并显示正确的数字是多少，再开始新游戏。请尝试编写这个版本的猜数游戏！

3.14.2　编程练习 2：显示和隐藏 Play Again 按钮

为本章的猜数游戏创建 GUI 时，我们决定不包含 Play Again 按钮，因为这个按钮仅在一轮游戏结束时才用得着，而在其他时候它只会让界面变得混乱。实际上，我们可使用方法 setVisible(false)将 GUI 组件隐藏起来，并在必要时使用 setVisible(true)重新显示它，就像处理 JFrame 时所做的。

首先，可在 GUI 布局中添加一个 Play Again 按钮，并将其命名为 btnPlayAgain；然后，在代码中调用 btnPlayAgain.setVisible(false)将这个按钮隐藏起来。在方法 checkGuess()中处理用户获胜的 else 语句中，可调用 call btnPlayAgain.setVisible(true)来显示这个按钮，而不直接开始新游戏。编写这些代码后，别忘了在 GUI 预览中双击这个按钮给它添加一个事件监听器，并在这个监听器中调用 newGame()，再将这个按钮隐藏起来。

需要指出的是，不同于另一个按钮（btnGuess），对于这个按钮，你需要在多个地方修改其 visible 属性以显示或隐藏它。与 txtGuess 和 lblOutput 一样，你需要在类开头声明私有的 JButton 变量 btnPlayAgain，同时避免在后面的代码中再次声明它。

如果你按前面的要求做了，Play Again 按钮将在游戏开始时被隐藏，等到用户获胜后才显示出来。用户单击该按钮时，将开始新游戏，而这个按钮将再次消失。请尝试玩玩包括 Play Again 按钮的游戏版本，如果你想到了其他改进用户体验的办法，去做就是了！

3.14.3 编程练习 3：创建 GUI 版 MadLib

回过头去看看你为完成第 2 章的编程练习 3 创建的程序 MadLibs.java，再创建一个新程序——MadLibGUI.java，它显示一个图形用户界面，其中包含让用户输入多个单词的标签和文本框，如 txtColor、txtPastTenseVerb、txtAdjective、txtNoun，还有一个用户可通过单击来生成 MadLibs 式故事的按钮。

请在 WindowBuilder Editor 中探索 GUI 元素的其他几个属性，如背景色、字体、初始文本值等。用户单击按钮时，这个程序将在一个 JLabel 或 JTextArea 中显示生成的 MadLibs 故事，如图 3-23 所示。

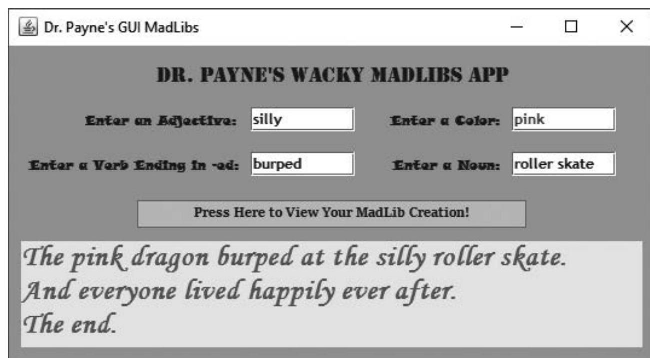
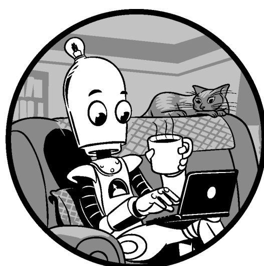


图 3-23 GUI 版 MadLibs 应用程序

第 4 章

创建你的第一个 Android 应用



为创建移动版猜数游戏，我们将使用 Android Studio。如果你还没有安装 Android Studio，请按 1.5 节的说明安装它。与 GUI 版一样，我们也将为移动版创建用户界面，如图 4-1 所示。本书前面创建的应用程序都只能在台式机上运行，但 Android 应用可在任何 Android 设备上运行，包括手机、平板电脑、手表、电视等。如果你没有 Android 设备，也不用担心，你依然可以开发 Android 应用，因为 Android Studio 自带了 Android 模拟器，让你能够模拟应用在实际设备上运行的情况。

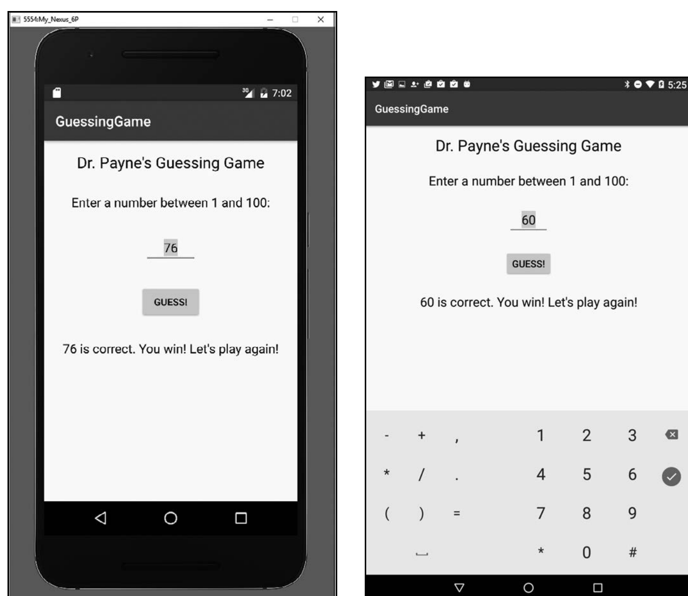


图 4-1　本章将开发的移动版猜数游戏在模拟的 Android 手机（左）和实际的 Android 平板电脑（右）上的运行情况

图 4-1 显示了本章要开发的应用在 Android 模拟器（模拟的是 Nexus 6P）和 Android 平板电脑上的运行情况。在这两种设备上运行的是同样的代码，更方便的是，我们可重用桌面版猜数游戏中的大量代码，因为 Android 是基于 Java 的！

注意，模拟器和实际设备看起来很像；虽然存在一些细微的差别，但这都是因为手机和 7 英寸平板电脑的屏幕尺寸不同导致的。为 Windows 系统编写的 Java 程序可在 macOS 和 Linux 系统中运行，同样，使用 Java 编写的 Android 移动应用可在数千种 Android 设备上运行。下面开始编写你的第一个 Android 移动应用！

4.1 在 Android Studio 中新建项目

首次启动 Android Studio 时，它可能花几分钟的时间进行更新。Android Studio 启动后，你将看到类似于图 4-2 所示的屏幕。请选择选项 Start a new Android Studio project。

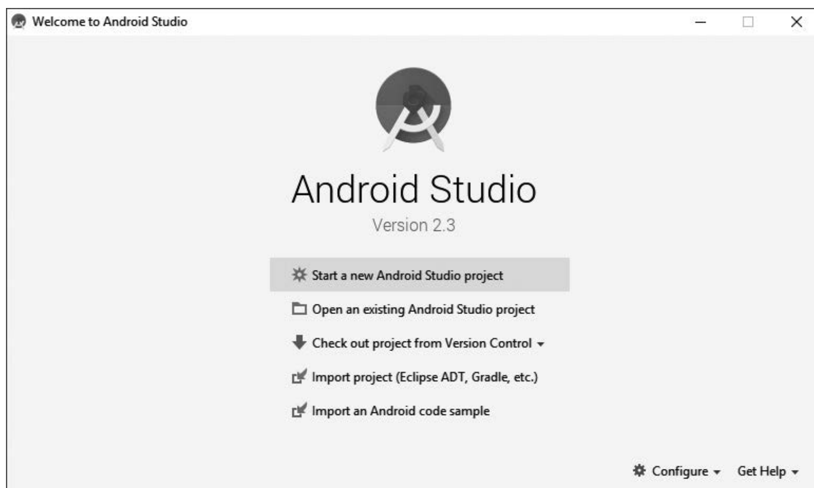


图 4-2 Android Studio 启动后，选择 Start a new Android Studio project

将新项目命名为 GuessingGame（注意中间没有空格）。如果你有网站，可在文本框 Company Domain 中输入它，如图 4-3 所示；否则保留其默认设置。接下来，选择项目存储位置。为确保组织有序，我创建了一个名为 AndroidProjects 的文件夹，你也应该这样做。

给项目命名后单击 Next 按钮。现在你可以指定应用要在哪个 Android 版本上运行。随着新设备和新功能的不断推出，Android 平台的发展速度很快，因此我们需要选择目标平台。所幸 Android Studio 会定期更新，这是基于 Android 新版本的推出情况以及有多少设备依然在使用旧版本进行的。选择要支持的 Android 版本时，还需选择**软件开发包**（SDK）或**应用编程接口**（API）等级。SDK 和 API 等级包含你要用来开发应用的工具，但它们与特定的 Android 版本相关联。Target Android Devices 窗口让你能够选择为应用选择最低的 SDK 或 API 等级，如图 4-4 所示。

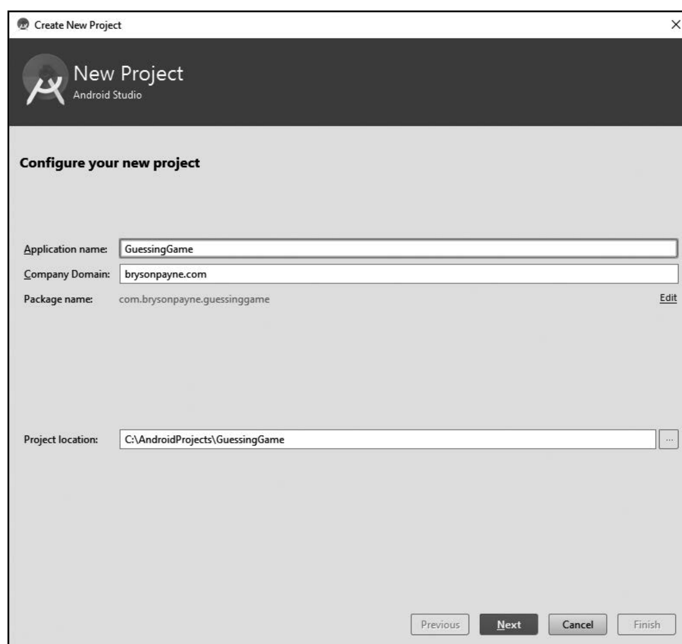


图 4-3　将 Android Studio 新项目命名为 GuessingGame（中间没有空格）

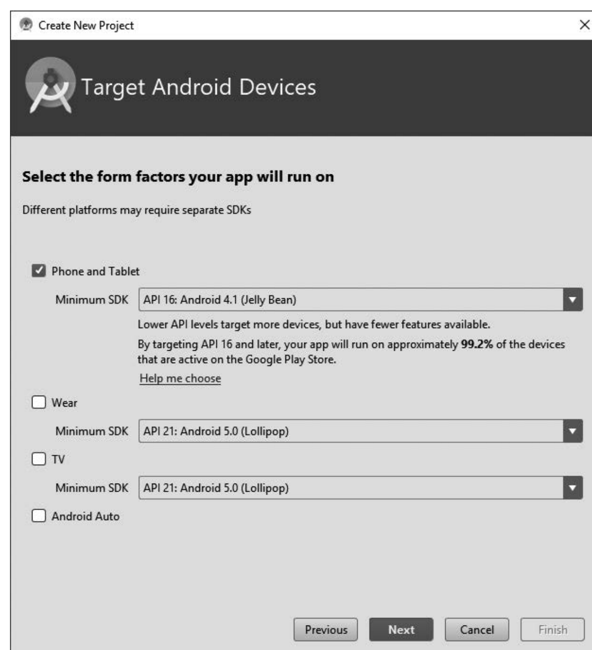


图 4-4　选择目标 Android 设备的最低 SDK 等级

对于猜数游戏，将最低 SDK 等级设置为 API 16（Android 4.1，Jelly Bean），Android Studio 将指出应用能够在超过 99%的 Android 设备上运行。单击 Next 按钮。

New Project 向导将询问你要在应用中添加哪种**活动**（activity）。活动就是用户可执行的操作；每个活动通常都有包含用户界面的屏幕布局，如图 4-5 所示。

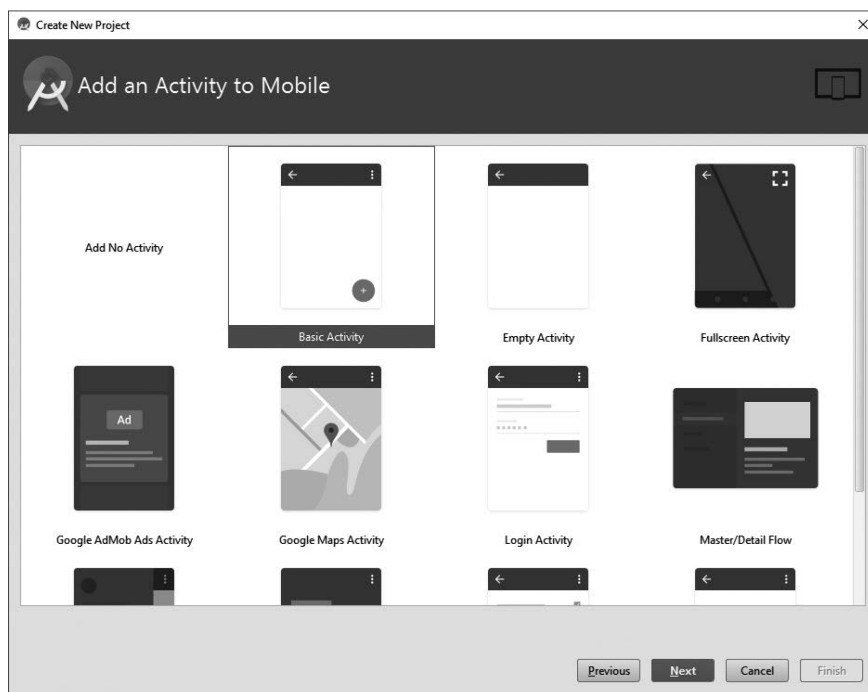


图 4-5 为应用选择 Basic Activity

可为应用选择的主活动很多，如 Google Maps、Login、Settings、Tabbed Activities 等。Basic Activity 将为猜数游戏打下坚实的基础，因此请选择它并单击 Next 按钮。

在接下来的屏幕中，使用 Android Studio 提供的默认名称：活动名 MainActivity 和布局名 activity_main，如图 4-6 所示。活动文件包含应用的代码；布局是一个独立的文件，包含应用的界面。单击 Finish 按钮结束项目创建工作。

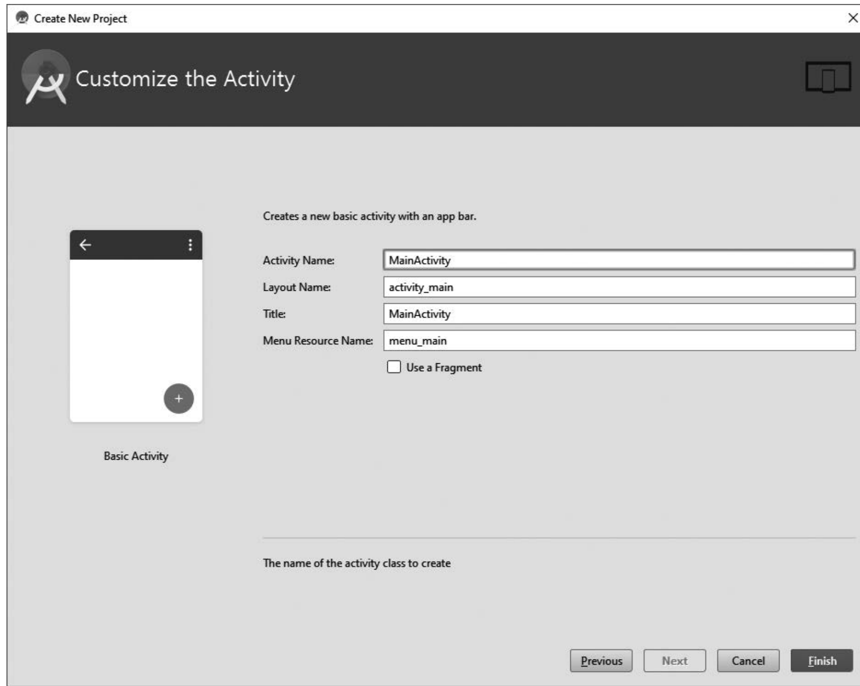


图 4-6　保留默认的活动名和布局名

构建过程可能需要一段时间，因为 Android Studio 需要创建项目文件并设置项目 Guessing Game。加载完毕后，Android Studio 将在默认视图下打开项目，如图 4-7 所示。

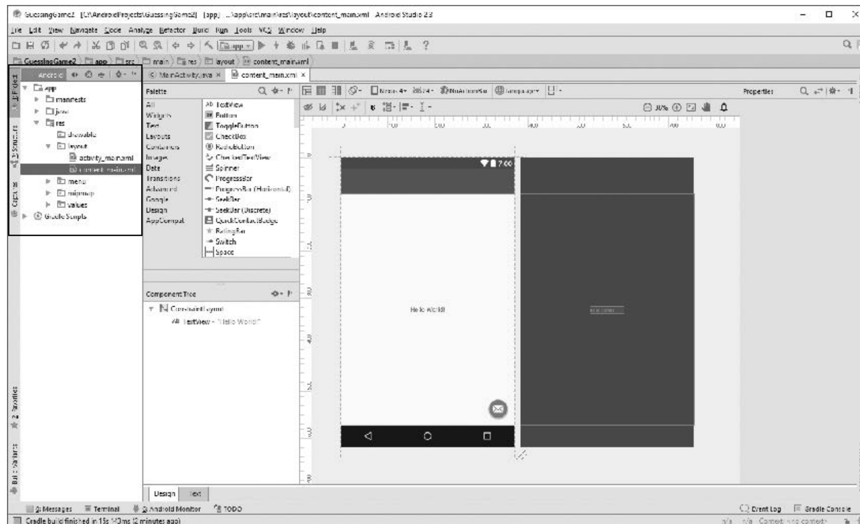


图 4-7　Android Studio 中的默认项目视图，其中包含一个 GUI 移动布局视图

如果你看到的视图不是这样的，请单击左上角的 Project 选项卡，再在 Project Explorer 中依次展开文件夹 app、res（表示**资源**）和 layout，再双击文件 content_main.xml，你将看到类似于图 4-7 所示的视图。

4.2　在设计视图中创建 GUI 布局

Android 应用将布局和代码（活动）分开，因此这个应用与第 3 章编写的桌面应用程序稍有不同。布局是使用**可扩展的标记语言**（eXtensible Markup Language，XML）而不是 Java 定义的。所幸虽然存在这种差别，我们依然可像使用 Eclipse 的 WindowBuilder Editor 那样通过拖放来设计 GUI 视图。在 Android Studio 中，主要不同在于组件名打上了移动应用的烙印。

与 WindowBuilder Editor 一样，Android Studio 也有两个选项卡，一个显示设计视图，一个显示源代码。在包含文件 content_main.xml 的主窗口中，单击左下角的 Design 选项卡，你将看到如图 4-8 所示的设计视图。

4

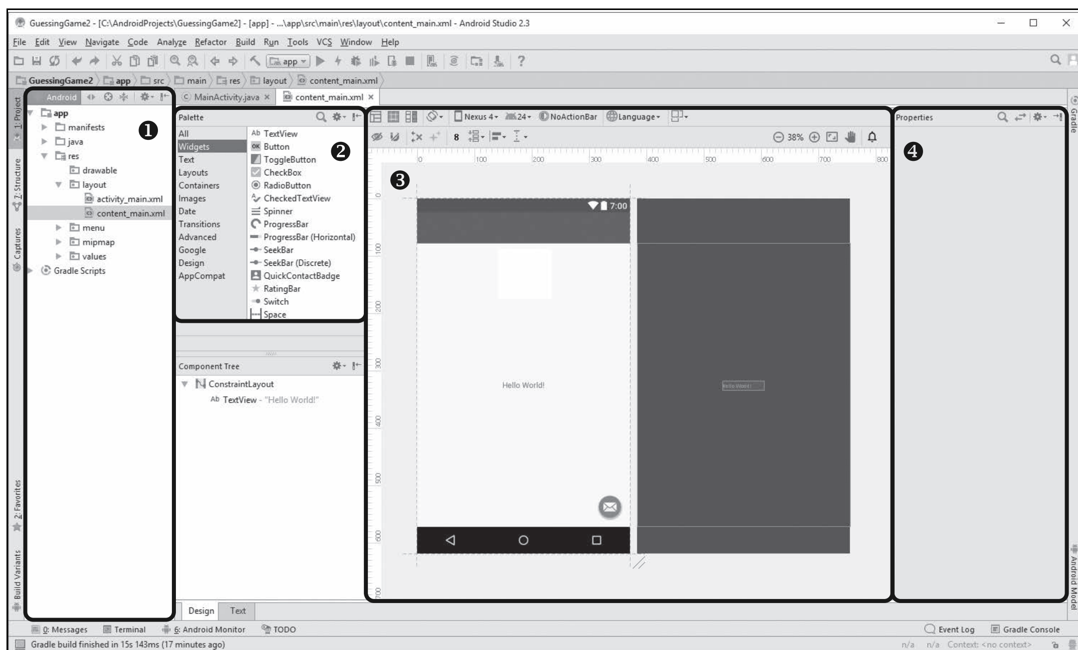


图 4-8　Android Studio 包含一个设计视图，该视图与 Eclipse 中 WindowBuilder Editor 包含的设计视图类似

图 4-8 显示了设计视图中最常用的 4 个区域：Project Explorer 面板❶、Palette 面板❷、Preview 面板❸和 Properties 面板❹。[①]

① 本书使用的是 Android Studio 2.3，界面随 Android Studio 版本的不同有所差异。——译者注

在 Preview 面板中央，有一个内容为“Hello World!”的标签——在 Android 中被称为 `TextView`。请单击这个标签并按 DELETE 键（也可右击该标签并选择 Delete）将其从 Preview 面板中删除。然后，单击 Palette 面板中的 Layouts 选项并选择 RelativeLayout，再将其拖曳到 Preview 面板或 Palette 面板下方的 Component Tree 中的 ConstraintLayout 上（请参见图 4-8）。我们将从一个空 RelativeLayout 开始创建猜数游戏的 GUI 视图。

在 Palette 面板中，选择 Widgets 中的 TextView。Android 中的 `TextView` 类似于 Swing 工具包中的 `JLabel`，而 Android 中的大部分 GUI 组件都被称为**控件**（widget）。我们将通过拖放将 GUI 组件放置到设计预览中。

单击控件 `TextView`，并将其拖曳到模拟 Android 手机内的白色应用背景上，将其放在顶部附近，以用作应用标题。

放置用作标题的 `TextView` 后，你将在右边的 Properties 面板中看到一些选项。如果这个面板没有出现，请单击 Design 选项卡右上角的字样 Properties，将这个面板展开。找到刚添加的 `TextView` 控件的 `text` 属性，将其改为 *`Your Name`*`'s Guessing Game`，再将 `textAppearance` 属性改为 `Large`。然后，拖曳这个 `TextView`，使其在屏幕顶部居中，如图 4-9 所示。灰色虚线参考线可帮助你将控件放到正确的位置。

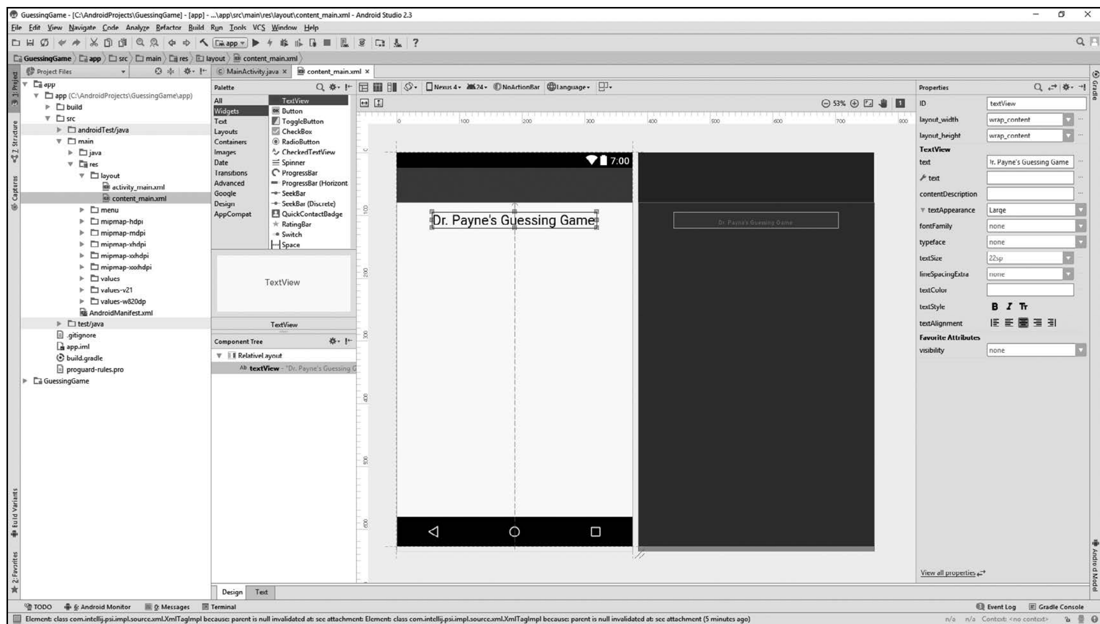


图 4-9　在 Android Studio 中添加 GUI 控件/组件的方式与 Eclipse 中很像

为添加提示用户输入 1~100 的数字的标签，再将一个 `TextView` 控件放到设计预览中，使其位于标题下方不远处。在右边的 Properties 面板中，将 `textAppearance` 属性设置为 `Medium`，再将 `text` 属性设置为 `Enter a number between 1 and 100`。

接下来，我们将添加让用户输入猜测的文本框，这种控件在 Android 中被称为 EditText。在 Palette 面板中，文本框位于 Widgets 中。在 Android 中，有很多不同类型的文本框，它们的行为各不相同，具体该使用哪种取决于应用的需求。例如，Plain Text 是简单文本框，Password 会隐藏用户输入的字符，Phone 显示数字键盘并将输入设置为电话号码格式，而 E-mail 显示包含符号@的键盘。

请选择名称为 Number 的文本框，这样用户将通过屏幕上的数字键盘来输入猜测。将这个文本框放在前一个标签下方，并使其与该标签的距离同该标签与第一个标签的距离相等。然后，在这个文本框下方添加一个 Button 控件，使其与文本框的距离同其他几个控件之间的距离相等，再将其 text 属性设置为 Guess!。最后，在按钮下方添加一个 TextView 控件，并将其 text 属性设置为 Enter a number,thenclickGuess!。如果无法让这个控件与按钮的距离同其他几个控件之间的距离相等（原因是它离屏幕中央太近），可在 Properties 面板中将 Layout_Margin 下的 layout_marginTop 设置为 30dp（你可能需要单击 View all properties，再单击 Layout_Margin 旁边的灰色箭头将其展开）。最终的 GUI 布局类似于图 4-10 所示。

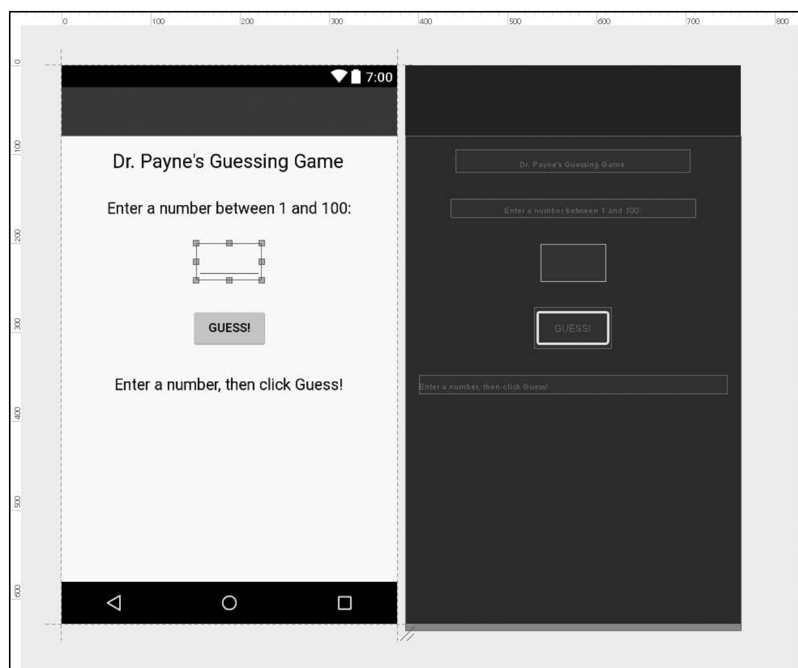


图 4-10 设计视图显示的 Android 猜数游戏的最终 GUI 布局

下面来做最后一项修改，然后给 GUI 组件命名。将供用户输入猜测的数字文本框的 width 属性改为 75，为此可单击 Properties 面板中的 View all properties 或左右箭头图标。对用户将在这个猜数游戏中输入的 1~100 的数字来说，这个文本框太宽，因此我们将其缩小得更窄些。

完成这个小小的修改后，该给 GUI 组件命名以便能够在 Java 代码中轻松地找到它们。

4.3　在 Android Studio 中给 GUI 组件命名

前一章在 Eclipse 中给 GUI 组件命名了，同样，在 Android Studio 中也需要给 GUI 元素命名，以便能够在代码中访问它们。然而，在 Android Studio 中，GUI 元素的名称默认不会出现在 Java 源代码中。相反，我们必须手动将这些 XML GUI 组件关联到 Java 代码，然后才能在代码中使用它们。

但就目前而言，我们只需将数字文本框、猜测按钮和最下面的标签的 `id` 属性分别改为 `txtGuess`、`btnGuess` 和 `lblOutput`。如果出现对话框 Update Usages，请单击 Yes 按钮。我们使用这些名称是出于一致性和方便考虑。图 4-11 显示了重命名后的文本框、按钮和标签。

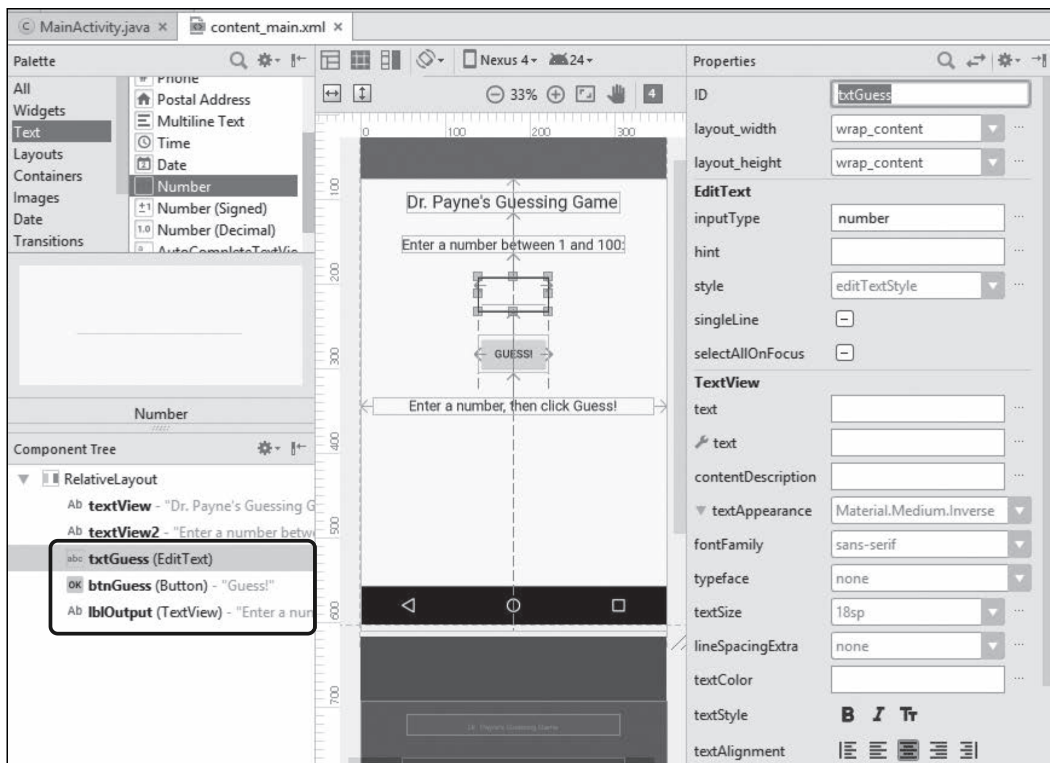


图 4-11　重命名这里圈出来了的文本框、按钮和标签：将它们的 `id` 属性分别改为 `txtGuess`、`btnGuess` 和 `lblOutput`

着手使用 Java 编写代码前，需要做的最后一件事情是隐藏小型的**浮动操作按钮**（fab）图标，如图 4-12 中设计预览的右下角所示。在你的项目中，可能没有这个 fab 图标，但如果有，请在 Project Explorer 面板中双击文件 activity_main.xml❶，再单击设计视图上方的 activity_main.xml 选项卡❷，然后单击 fab 图标❸，并在 Properties 面板中将 `visibility` 属性设置为 `invisible`❹。

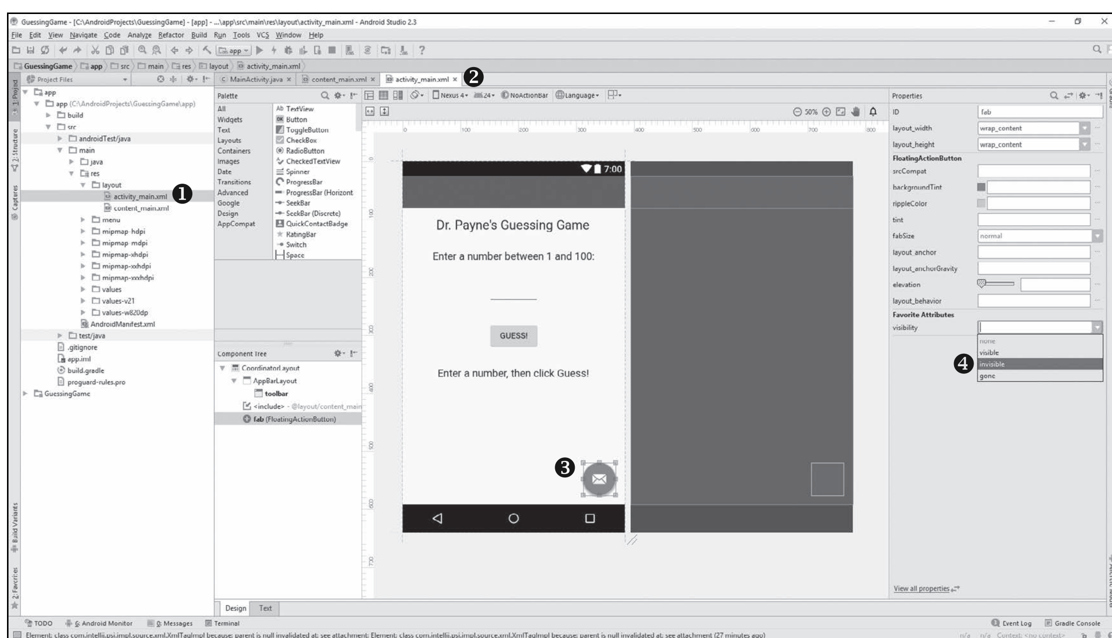


图 4-12　隐藏 fab 图标：打开文件 activity_main.xml，单击小型信封图标，并将 `visibility` 属性设置为 `invisible`

这里让 fab 图标不可见，而没有删除它，因为你在后面可能想使用它在应用中添加新功能。浮动操作按钮可用来将信息分享到 Facebook、在 Twitter 发布有关你有多喜欢这个应用的推文、向朋友发送有关该应用的 E-mail 等。

接下来，把这个 GUI 关联到 Java 代码，以便在 Java 中使用它来完成这个应用的编程工作。

4.4　在 Android Studio 中将 GUI 关联到 Java 代码

现在该将 GUI 关联到 Java 以便为猜数游戏编写代码了。首先，打开 Java 源代码文件；为此，在 Project Explorer 面板中，依次展开 app、src、main、java、com.example.guessinggame（或你指定的包名），再双击 MainActivity 打开源代码文件 MainActivity.java。

你可能注意到了，在 Project Explorer 面板中，文件夹 java 下还有其他的包，它们的名称也以 com.example.guessinggame 打头，但后面还有(test)或(androidTest)。这些包用于测试，让工程师能够评估应用的品质、安全性和功能，以保证软件质量。我们不会讨论品质保证，但这是进入软件开发行业的绝佳途径。

文件 MainActivity.java 中的 Java 源代码类似于图 4-13 所示。

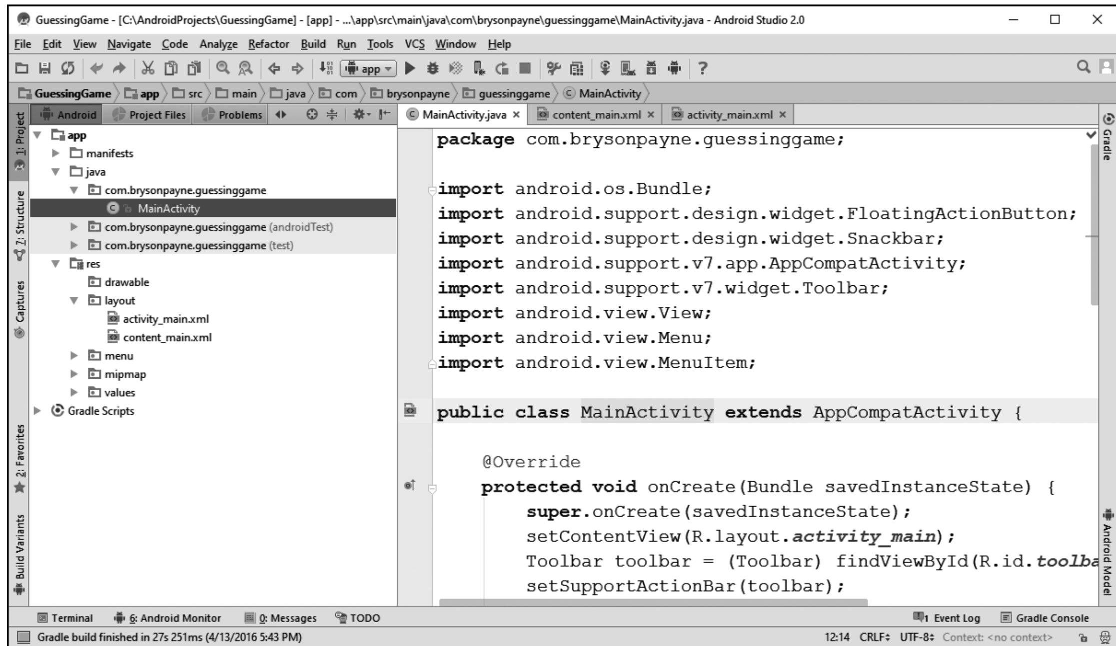


图 4-13　应用的 Java 源代码文件默认为 MainActivity.java

请注意，在 Android Studio 中，源代码文件的开头是一个包声明。在复杂的程序（如移动应用）中，包可帮助我们组织所有必要的文件。在这里，包名是在 Create New Project 对话框中指定的公司域名（参见图 4-3），但顺序相反——com 在最前面。

包声明后面是多条 Android `import` 语句，它们的作用与 Java 桌面应用中相同——导入既有的特性和功能。

接下来的 `public class MainActivity` 代码片段可能随你选择的最低 API 等级而稍有不同，但总体而言是相似的，且我们将编写的应用适用于多个 API 等级。首先，我们将声明将 GUI 与程序相关联的变量。请在 `MainActivity` 类的左大括号下一行单击，并添加如下代码行，以声明表示文本框、按钮和输出标签的变量：

```
public class MainActivity extends AppCompatActivity {
    private EditText txtGuess;
    private Button btnGuess;
    private TextView lblOutput;
```

当你输入变量的类型时，可能出现一个列表让你能够导入所需的类型（如 `android.widget.EditText`）。如果你双击要导入的控件类型，Android Studio 将为你添加相应的 `import` 语句。如果你输入这 3 行代码时，没有通过双击来接受自动导入，只需在每种控件类型上单击，再按 Alt-回车（在 macOS 系统中为 Option-回车），就可导入相应的类，如图 4-14 所示。

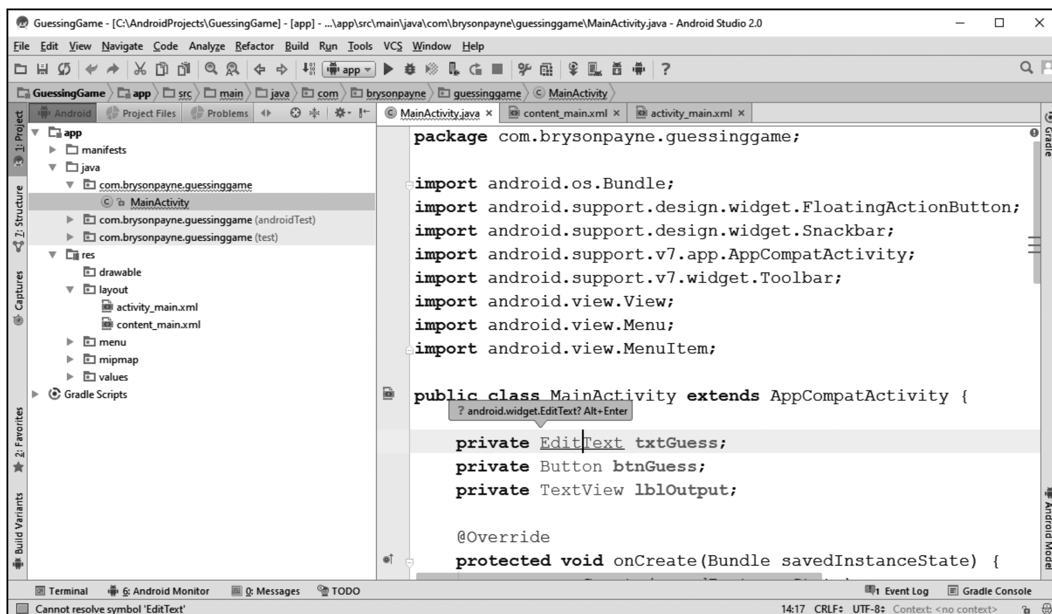


图 4-14　与 Eclipse 中一样，可让 Android Studio 自动导入类；为此可在输入时双击，也可在输入后使用简单组合键 Alt-回车（在 macOS 系统中为 Option-回车）

现在，下面 3 条 import 语句将与其他 import 语句一起出现在文件开头附近：

```
import android.widget.EditText;
import android.widget.Button;
import android.widget.TextView;
```

声明这 3 个表示 GUI 控件的变量后，需要将它们关联到相应的 XML 控件。我们将在应用程序加载时运行的方法 onCreate()中完成这项任务，如图 4-14 底部所示。Android Studio 自动为这个方法生成了一些代码。

请在方法 onCreate()的开头找到下面这个代码行：

```
    setContentView(R.layout.activity_main);
```

将光标放在这行代码末尾，按回车两次，再输入下述不完整的代码行，着手将变量 txtGuess 关联到 XML 布局文件中的 EditText 控件：

```
    txtGuess = (EditText) findViewById(R.id.
```

函数 findViewById()用于将 XML 布局中的 GUI 控件关联到我们要在源代码中用来表示它们的变量。这个函数内的 R 表示 Android Studio 生成的特殊文件 R.java，这个文件让你能够关联到资源。R 是资源的缩写，通常存储在项目的 res 文件夹中。输入上述不完整的代码行后，你将看

到一个类似于图 4-15 所示的列表，请在其中找到 txtGuess 并双击它。如果找不到选项 txtGuess，请返回到 content_main.xml 的设计视图，确认将这个文本框的 id 属性设置成了 txtGuess。

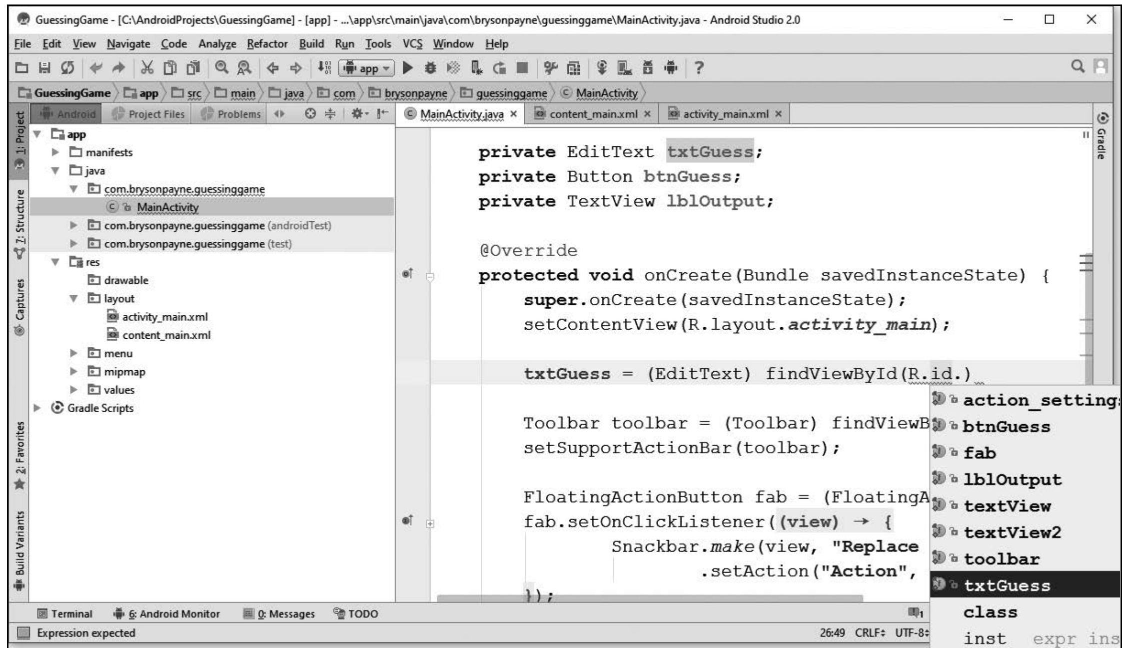


图 4-15　Android Studio 显示一个列表，帮助你将 Java 代码关联到布局中的 GUI 资源

最后，加上右括号和分号，如下面的代码清单所示。接下来，关联按钮和输出标签。你要在方法 onCreate()中添加的 3 行代码如下所示：

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    txtGuess = (EditText) findViewById(R.id.txtGuess);
    btnGuess = (Button) findViewById(R.id.btnGuess);
    lblOutput = (TextView) findViewById(R.id.lblOutput);
```

如果你在布局中给控件都指定了正确的名称，这些代码行将把变量 txtGuess、btnGuess 和 lblOutput 分别关联到 GUI 布局中的 EditText、Button 和 TextView 组件。现在请保存对项目所做的所有修改。

4.5　添加检查猜测及开始新游戏的方法

下面来编写方法 checkGuess()。我们可将方法 checkGuess()放在变量 txtGuess、btnGuess、lblOutput 的声明和方法 onCreate()之间，如图 4-16 所示。

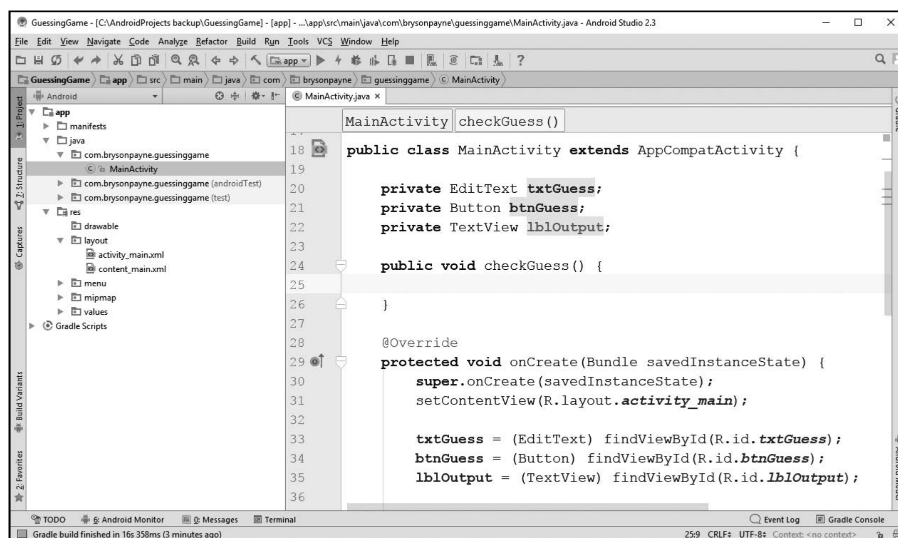


图 4-16 着手编写方法 checkGuess()

首先，需要从 GUI 文本框中获取用户的猜测，并将其存储在一个 String 变量中：

```
public class MainActivity extends AppCompatActivity {
    private EditText txtGuess;
    private Button btnGuess;
    private TextView lblOutput;
    public void checkGuess() {
        String guessText = txtGuess.getText().toString();
```

从文本框获取用户猜测的代码几乎与桌面版中完全相同，只是在末尾添加了方法.toString()。Android 中有一个 Text 类，而用户在文本框中输入的文本是一个 Text 对象，因此我们必须将这个对象转换为 String 对象。

所幸方法 checkGuess()的其他代码可直接从第 3 章的桌面 GUI 版中复制并粘贴，而无需做任何修改！完整的 checkGuess()方法类似于下面这样：

```
public void checkGuess() {
  ❶ String guessText = txtGuess.getText().toString();
    String message = "";
  ❷ try {
      ❸ int guess = Integer.parseInt(guessText);
        if (guess < theNumber)
            message = guess + " is too low. Try again.";
        else if (guess > theNumber)
            message = guess + " is too high. Try again.";
        else {
            message = guess +
                    " is correct. You win! Let's play again!";
            newGame();
        }
  ❹ } catch (Exception e) {
```

```
        message = "Enter a whole number between 1 and 100.";
  ❺ } finally {
        lblOutput.setText(message);
        txtGuess.requestFocus();
        txtGuess.selectAll();
    }
}
```

Java 的优点之一是可跨平台重用代码。创建桌面 GUI 游戏时，我们使用了命令行游戏中的代码，同样，在 Android 版游戏中，我们可使用桌面版中的代码。下面复习一下这些代码，以理解其在 Android 版中的工作原理。

在❶处，我们从文本框中获取用户的猜测，并创建表示输出消息的字符串。在❷处，我们着手编写一条处理用户输入错误（异常）的 try-catch-finally 语句。在 try 块中，我们从用户输入的字符中提取用户猜测的整数❸，再使用 if-else 语句检查用户猜大了、猜小了还是猜对了，并相应地设置消息；同时在用户猜对了时开始新游戏。接下来，catch 语句❹让用户输入 1~100 的整数。然后，finally 块❺将控件 lblOutput 的 text 属性设置为 message，并为用户再次输入做好准备：将光标放到文本框中并选择所有的文本。

你可能注意到了，有两项内容下面带红线——theNumber 和 newGame()，这是因为我们还没有定义它们。

请在 MainActivity 类的开头（3 个 GUI 控件声明后面），添加表示秘密数字的变量 theNumber 的声明：

```
public class MainActivity extends AppCompatActivity {
    private EditText txtGuess;
    private Button btnGuess;
    private TextView lblOutput;
    private int theNumber;
```

这样，代码与桌面版猜数游戏中相同，因为无论是命令行版、桌面版还是 Android 移动版中，声明 int 变量的 Java 代码相同。

方法 newGame()的代码也与桌面版中相同。请在方法 onCreate()前面添加方法 newGame()：

```
    public void newGame() {
        theNumber = (int)(Math.random() * 100 + 1);
    }
@Override
protected void onCreate(Bundle savedInstanceState) {
```

在 Android 版中，开始新游戏的方式与桌面版相同：使用函数 Math.random()将变量 theNumber 设置为一个 1~100 的随机数。

我们可以在方法 onCreate()中（关联 3 个 GUI 组件的代码后面）调用 newGame()：

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
```

```
txtGuess = (EditText) findViewById(R.id.txtGuess);
btnGuess = (Button) findViewById(R.id.btnGuess);
lblOutput = (TextView) findViewById(R.id.lblOutput);
newGame();
```

这个游戏启动时，将调用方法 newGame()来选择一个要让用户去猜测的随机数。接下来，我们要处理按钮单击事件，并学习如何在 Android 设备和 Android 模拟器上运行应用。

4.6　在 Android 中处理事件

在 Eclipse 中，要给猜测按钮添加事件监听器，只需在设计视图中双击它。遗憾的是，在 Android Studio 中没那么容易，因为 GUI 布局和源代码是分开的。所幸 Android Studio 提供了**代码补全**选项（类似于 Eclipse 中的内容助手功能），可帮助我们添加事件监听器。

为添加按钮单击监听器，请在方法 onCreate()中（调用 newGame()的代码后面）输入下述不完整的代码行：

```
txtGuess = (EditText) findViewById(R.id.txtGuess);
btnGuess = (Button) findViewById(R.id.btnGuess);
lblOutput = (TextView) findViewById(R.id.lblOutput);
newGame();
btnGuess.setOn
```

Android Studio 的代码补全功能将显示一个代码推荐列表，如图 4-17 所示。请双击这个列表中的 setOnClickListener()，将其添加到程序中。

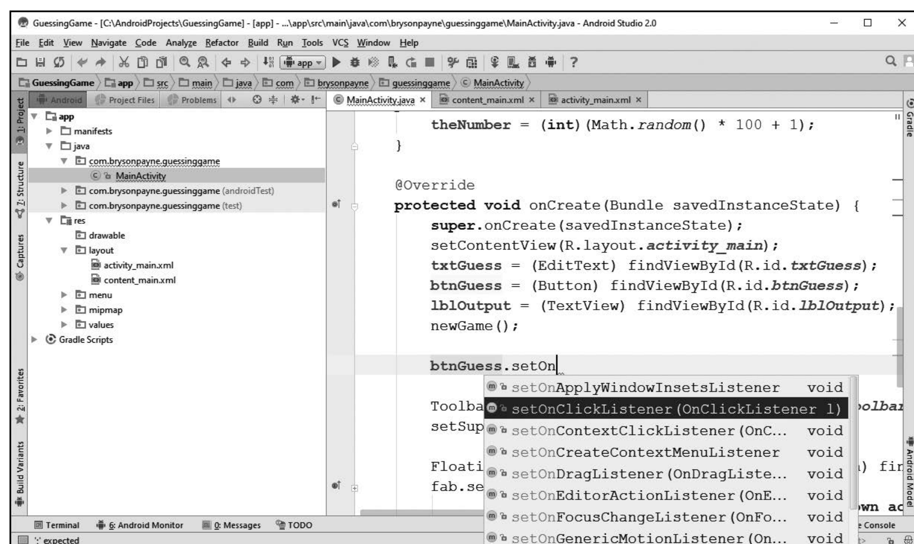


图 4-17　当你输入代码时，Android Studio 的代码补全功能会提供建议，就像 Eclipse 的内容助手功能一样

在 btnGuess.setOnClickListener()的括号内，输入 new 和 OnClickListener，Android Studio 的代码补全功能将显示一个列表，如图 4-18 所示。

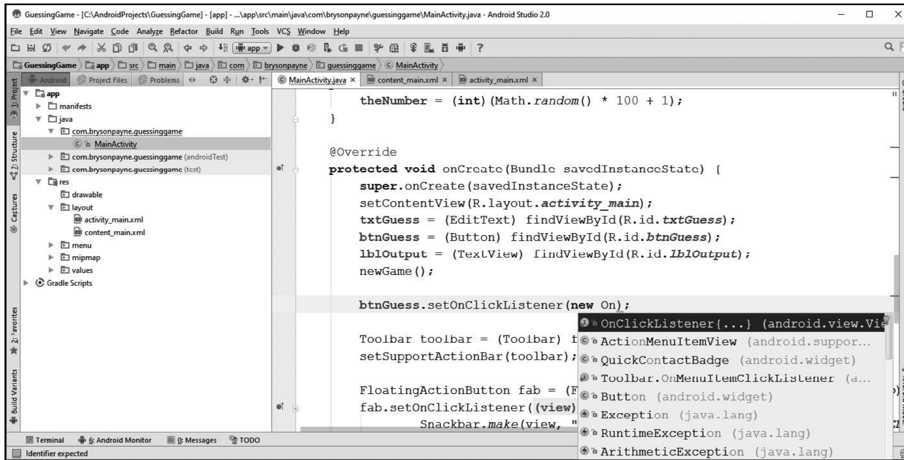


图 4-18　使用代码补全功能给按钮 btnGuess 添加 OnClickListener

双击该列表中的 OnClickListener，Android Studio 将添加几行代码。此时，按钮 btnGuess 的事件监听器的代码类似于下面这样：

```
btnGuess.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
    }
});
```

似曾相识？这也是一个内部匿名类。在桌面版中，也有一个内部匿名类，但名称稍有不同。这两个内部匿名类的作用几乎相同。你可能注意到了，Android Studio 在多个地方都插入了 @Override，这是一个**编译器指令**，告诉编译器你要重写父类的方法。

我们要在用户单击按钮 Guess!时将其猜测与秘密数字进行比较，因此在方法 onClick()的大括号内添加代码 checkGuess();。

```
public void onClick(View v) {
    checkGuess();
}
```

至此，移动版猜数游戏已创建好，可以运行了。该应用的完整代码如下所示。你的版本可能稍有不同——可能还包含另外两个处理菜单项的方法，但由于我们没有使用菜单，这里将它们删除了以节省篇幅（有关如何创建选项菜单和设置，请参阅第 5 章）：

```
package com.brysonpayne.guessinggame;
import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
```

```
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.EditText;
import android.widget.Button;
import android.widget.TextView;
import org.w3c.dom.Text;
public class MainActivity extends AppCompatActivity {
    private EditText txtGuess;
    private Button btnGuess;
    private TextView lblOutput;
    private int theNumber;
    public void checkGuess() {
        String guessText = txtGuess.getText().toString();
        String message = "";
        try {
            int guess = Integer.parseInt(guessText);
            if (guess < theNumber)
                message = guess + " is too low. Try again.";
            else if (guess > theNumber)
                message = guess + " is too high. Try again.";
            else {
                message = guess +
                        " is correct. You win! Let's play again!";
                newGame();
            }
          } catch (Exception e) {
            message = "Enter a whole number between 1 and 100.";
          } finally {
            lblOutput.setText(message);
            txtGuess.requestFocus();
            txtGuess.selectAll();
          }
      }
      public void newGame() {
          theNumber = (int)(Math.random() * 100 + 1);
      }
      @Override
      protected void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.activity_main);
          txtGuess = (EditText) findViewById(R.id.txtGuess);
          btnGuess = (Button) findViewById(R.id.btnGuess);
          lblOutput = (TextView) findViewById(R.id.lblOutput);
          newGame();
          btnGuess.setOnClickListener(new View.OnClickListener() {
              @Override
              public void onClick(View v) {
                  checkGuess();
              }
          });
          Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
          setSupportActionBar(toolbar);
      }
}
```

在接下来的两节中，你将学习如何在 Android 模拟器和 Andriod 设备上运行这个应用。

4.7　在 Android 模拟器中运行应用

至此，我们编写了应用所需的 Java 代码，但还需进行测试。不同于命令行和桌面应用，移动应用无法在 PC 上运行，因为 PC 没有 Android 操作系统。要运行移动应用以便对其进行测试，需要有 Android 设备或在 PC 上模拟 Android 设备的模拟器。本节将创建一个 Android 虚拟设备，用于测试应用。

在当前显示的是文件 MainActivity.java 的情况下，单击 Run 按钮或选择菜单 Run▶Run ‘app’，将出现一个弹出窗口，让你选择**部署目标**（deployment target），即要在其中运行应用的设备，如图 4-19 所示。

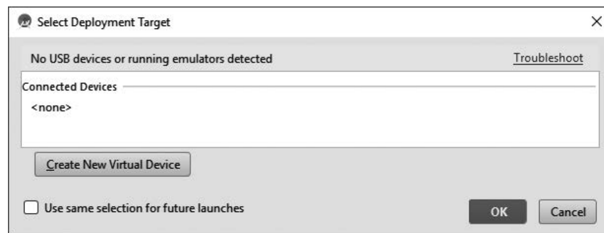


图 4-19　要运行 Android 应用，必须选择目标设备——模拟器或实际设备

单击 Create New Virtual Device 按钮以配置新的 **Android 虚拟设备**（AVD）。请先选择设备类型，如图 4-20 所示，再单击 Next 按钮。

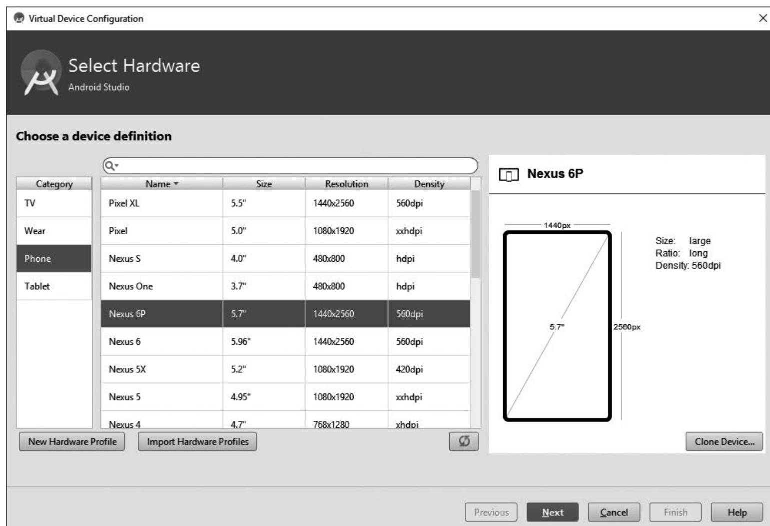


图 4-20　选择要模拟的设备

这里选择 Nexus 6P，但你可随便选择其他设备。如果应用要发布到 Google Play Store，可能要在屏幕尺寸和类型各异的设备上进行测试，因此可创建多个模拟器。但我们先从小处着手，只创建一个模拟器。

接下来，需要选择系统映像——通常为 x86 或 ARM。我们要使用一个 ARM 映像，因此单击选项卡 Other Images，如图 4-21 所示。

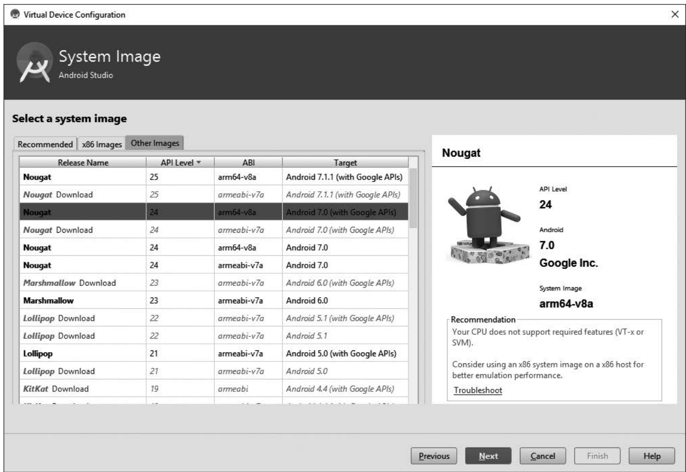


图 4-21　可选择 x86 或 ARM 模拟器；ARM 的速度更慢，但通常在不同处理器中的行为更一致

如果你要选择的 Android 版本、API 等级或模拟器呈灰色，请单击相应的 Download 链接。

选择所需的 Android 模拟器后，单击 Next 按钮。在这里，我们选择带 Google API 的 Nougat API 24 ARM Android 7.0。你将看到最后一个屏幕，让你确认配置，包括你要修改的高级设置。我们将这个 AVD（要模拟的设备）的名称改为 My Nexus 6P，如图 4-22 所示。

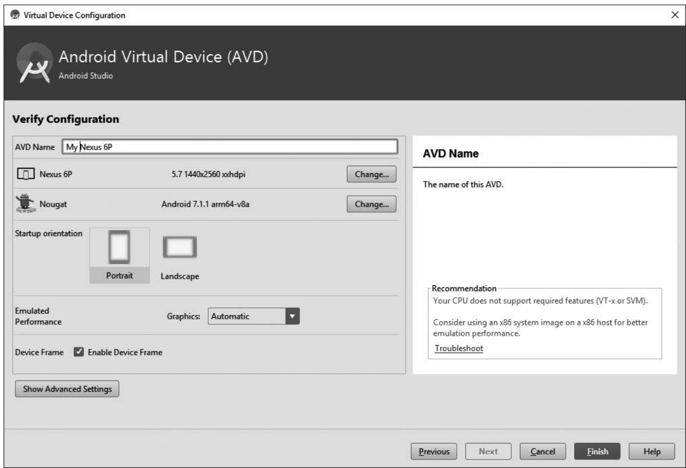


图 4-22　给新建的 AVD 命名

如果你的计算机较旧或内存少于 8GB，无法运行模拟器，请单击 Show AdvancedSettings 按钮，再向下滚动到 Memory and Storage 部分，将 RAM 量改为 768MB。

单击 Finish 按钮，将创建指定的 AVD 并将其保存到磁盘。现在该启动模拟器并尝试新创建的设备了。你将再次看到 Select Deployment Target 窗口，但现在其中包含模拟器 My Nexus 6P。请单击 OK 按钮，如图 4-23 所示。

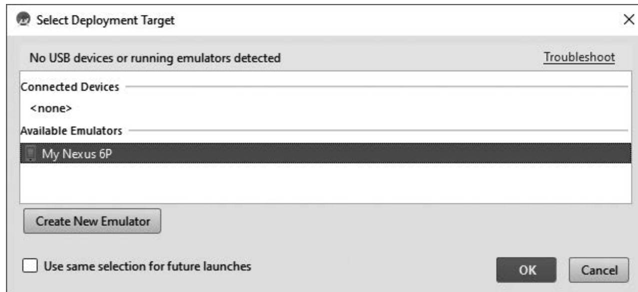


图 4-23　将刚创建的模拟器指定为部署目标

过段时间后，你将看到消息“Starting AVD”，然后是类似于 Android 启动屏幕的模拟器窗口，如图 4-24 所示。首次运行模拟器时，这可能需要几分钟。模拟器启动后，你可能会看到一个锁定屏幕。

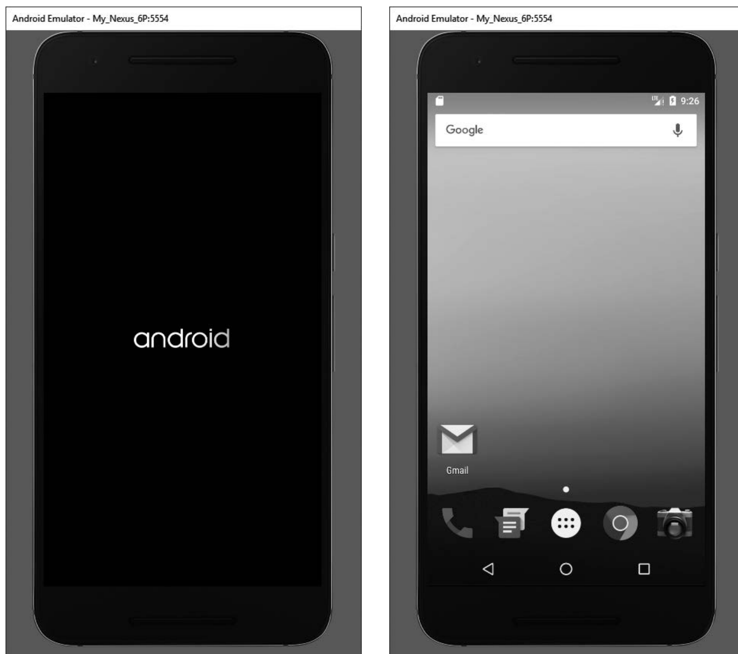


图 4-24　Android 模拟器正在启动（左）；Android 虚拟设备的主屏幕（右）

如果模拟器显示的是锁定屏幕，请单击屏幕底部的锁并向上拖曳来解锁虚拟设备，这类似于在 Android 设备上向上滑动来解锁。你可能看到一两个欢迎屏幕，但可通过单击来关闭这些屏幕，直到到达主屏幕，如图 4-24 所示。

现在，返回到 Android Studio 并再次单击 Run 按钮。这次你将在 Select Deployment Target 对话框中看到新创建的 AVD，如图 4-25 所示。请选择它并单击 OK 按钮。

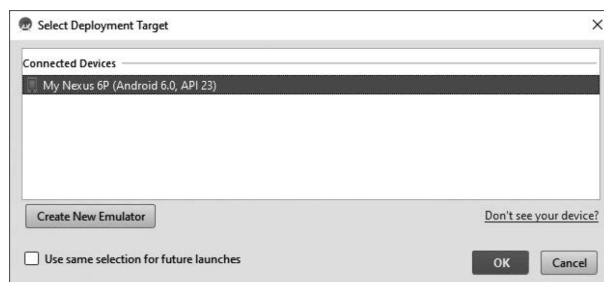


图 4-25　正在运行的模拟器出现在连接的设备列表中

这个项目将再构建一次，然后将其可执行版本传输到模拟器（首次运行模拟器时，可能询问你是否要更新 Android SDK Tools）。几分钟后，你将看到猜数游戏在模拟器中运行！

代码都已编写好，因此你应该能够在模拟器上玩这个游戏。图 4-26 显示了这个游戏的一整轮情况。

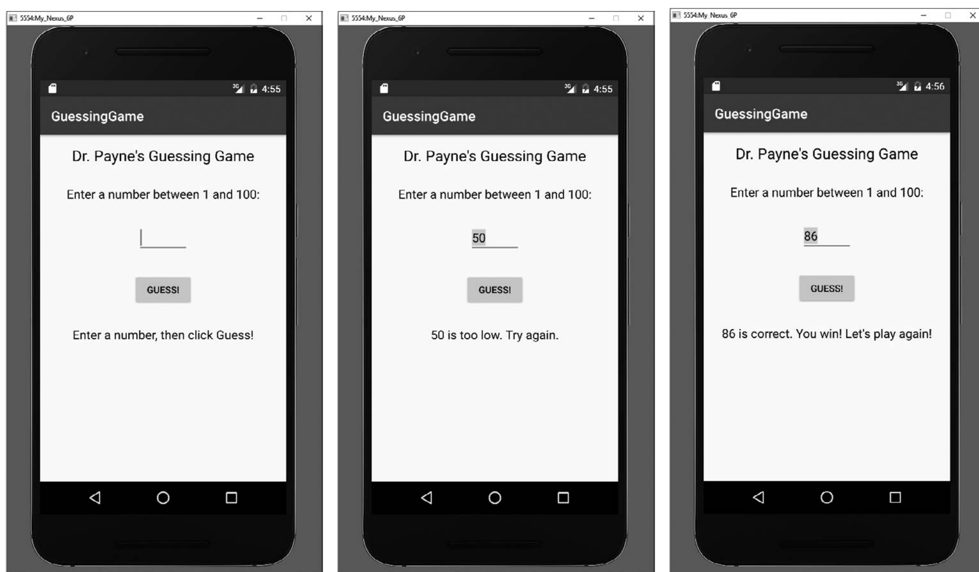


图 4-26　在模拟器中运行的猜数游戏（左）；用户猜测一次后（中）；用户获胜后（右）

其运行情况与桌面版相同，但运行在 Android 模拟器上。请通过键盘输入猜测，并使用鼠标

单击按钮 Guess!。与 GUI 桌面版一样，我们将从两个方面改进 Android 移动版的用户体验，但它目前是功能齐备的！

改善用户体验之前，我们将介绍如何在 Android 设备上运行这个应用。请不要关闭 Android 模拟器；不使用时可将其最小化——编程期间让它继续在后台运行，这样可避免漫长的启动过程。

注意　如果你没有 Android 设备，可直接跳到 4.9 节。

4.8　在 Android 设备上运行应用

要在 Android 设备上运行应用，需要花点时间做准备，但如果你使用 USB 电缆将设备连接到了计算机，准备工作很快就能完成。

4.8.1　准备好设备

要将应用部署到 Android 设备，必须在设备上启用“开发者”模式。你还需修改两项设置，以便能够开发和调试应用。

在你的 Android 设备上，点击“设置”，再向下滚动到末尾，以找到并点击“关于平板电脑”“关于手机”或“关于设备”。找到“版本号”并点击至少 7 次，以启用隐藏的开发者模式。在开发者模式下，可在设备上测试应用。图 4-27 显示了“设置”菜单（左）和启用开发者模式后的“关于”菜单（中）。

启用开发者模式后，单击“关于”屏幕左下角的返回箭头，再在“设置”屏幕中点击“开发者选项”。确保启用了“开发者选项”，如图 4-27（右）所示。你还应启用“USB 调试”。

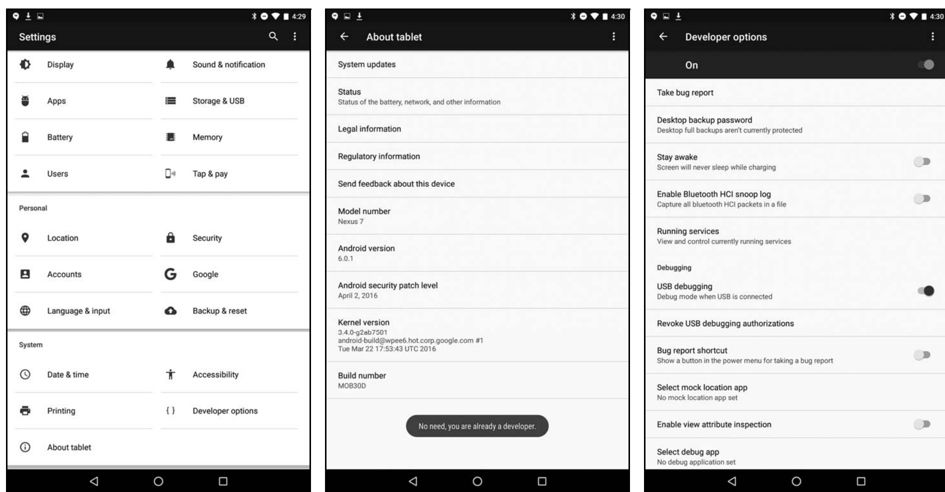


图 4-27　Android Nexus 7 平板电脑的“设置”菜单（左）、“关于”菜单（中）和“开发者选项”（右）

现在可以将 Android 手机、平板电脑或其他设备连接到计算机并运行应用了。

4.8.2 连接设备

要将 Android 设备连接到计算机，需要一条 USB 电缆，最好是手机或平板电脑自带的。这是一条微型 USB 电缆，通常随充电器一起提供。请注意，并非所有充电器电缆都是功能齐备的 USB 电缆：如果执行如下操作后不管用，请尝试换条电缆。

使用 USB 电缆将设备连接到计算机。插入电缆后，你的手机或平板电脑的屏幕上将出现一个类似于图 4-28 的窗口，询问你是否允许从计算机进行 USB 调试，请点击 OK 按钮。你可选择复选框 Always allow from this computer，这样你再通过 USB 电缆连接到计算机时，将不会出现这个窗口。

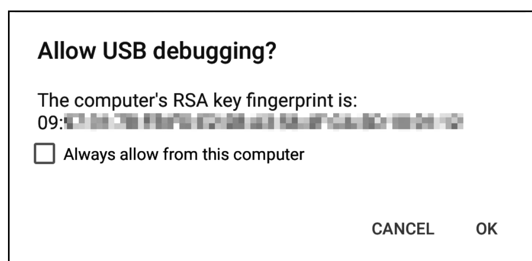


图 4-28 首次将 Android 设备连接到计算机时，计算机将询问你是否要允许 USB 调试

允许 USB 调试启用了两项功能。首先，可通过传输 Android 包文件（APK）将我们编写的应用传输到 Android 设备。相比于在模拟器上运行应用，这样做的速度要快得多。另外，在实际设备上测试应用的行为也是更佳的方式。其次，我们还能通过 USB 连接来调试应用，这意味着我们从 Android 设备获取信息来帮助调试。这些信息包括错误以及可在类似于控制台的应用程序（在 Android Studio 中，这个程序名为 logcat）中读取的日志项；它们类似于查找命令行和桌面 GUI 应用程序中的 bug 时使用的控制台输出。

将 Android 设备连接到计算机后，下面来看看如何在设备上运行猜数游戏。

4.8.3 在设备上运行应用

现在可以尝试在 Android 设备上运行应用了。在 Android Studio 中，单击 Run 按钮或选择菜单 Run ▸ Run ‘app’。这次 Select Deployment Target 窗口将显示两个连接的设备——模拟器（My Nexus 6P）和你的 Android 设备，如图 4-29 所示。

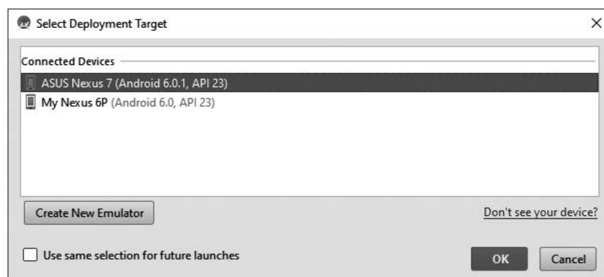


图 4-29　如果你启用了开发者模式和 USB 调试并将设备连接到了计算机，将能够选择在你的 Android 设备上运行应用

选择你的 Android 设备并单击 OK 按钮。Android Studio 可能需要花点时间来构建应用，但一旦它将 APK 文件传输到你的设备，你就将看到猜数游戏立即在你的平板电脑或手机上运行。请玩个一两轮，核实它像在模拟器上运行时一样好（可能更好，至少速度更快）。

你可能立即会发现一个重要的不同：屏幕底部有一个数字键盘，如图 4-30 所示。

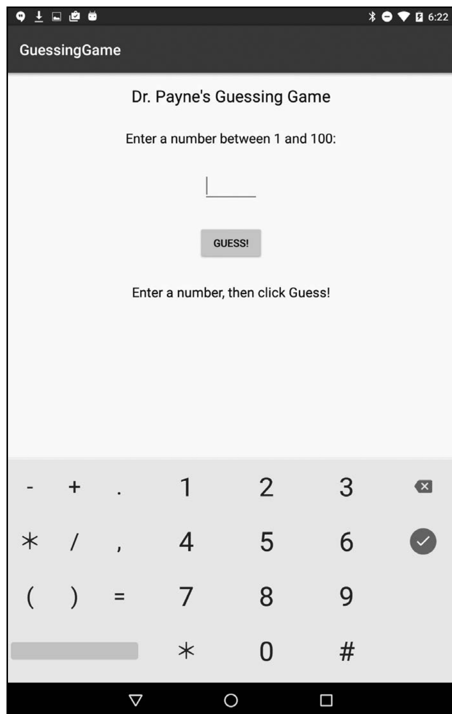


图 4-30　在 Android 设备上运行时，猜数游戏默认显示数字键盘

前面添加供用户输入猜测的文本框（`EditText`）时，选择的是 Number 文本框，因此光标位于该文本框中，将出现数字键盘。然而，由于模拟器运行在计算机上，没有自己的键盘，因此前

面在模拟器中运行这个应用时，你可能看不到数字键盘。

能够在 Android 手机、平板电脑和其他设备上运行猜数游戏后，下面来对它做最后几项修改，以改善用户体验。

4.9 改善用户体验

猜数游戏在 Android 模拟器和 Android 设备上都运行得很好，但在用户体验方面还有一些改进空间。首先，我们让用户在 Number 文本框中输入的数字居中；其次，我们让用户按回车时的效果与单击 Guess!按钮相同，让这个应用更直观。

4.9.1 让用户在文本框中输入的字数居中

在 Android Studio 中，再次打开布局文件 content_main.xml。单击设计预览中的文本框 `txtGuess` 以选择它。在 Properties 面板中找到属性 `textAlignment`。在 Android 中，这个属性类似于桌面 GUI 中的属性 `horizontalAlignment`，让我们能够修改用户输入的文本的对齐方式。

单击文本居中对齐选项——看起来像居中文本的图标。要想在不运行应用的情况下，在设计预览中测试这一点，可通过滚动找到属性 `text`，并将其设置为 `50`。在设计预览中，数字 50 将出现在文本框中央。如果你不想在该应用中显示这个数字，务必将其删除。

4.9.2 添加回车键监听器

下面来创建一个处理用户按回车键的事件监听器，它像处理用户单击 Guess!按钮那样做——调用 `checkGuess()`。

返回到源代码文件 MainActivity.java，向下滚动到方法 `onCreate()`，并找到给按钮添加 `onClickListener` 的代码（要看到像下面列出的代码，你可能需要单击 `btnGuess.setOnClickListener()`左边的加号，将这个事件监听器的代码展开）。在这些代码后面，给文本框 `txtGuess` 添加一个事件监听器，以处理按回车键等操作事件，如下所示：

```
btnGuess.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        checkGuess();
    }
});
txtGuess.setOnEditorActionListener(new TextView.OnEditorActionListener() {
    @Override
    public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
        checkGuess();
      ❶ return true;
    }
});
```

这里监听的**编辑器操作**（editor action）是用户在文本框中输入时按回车键。在这个事件发生

时，我们要检查用户输入的猜测。

我们让 return 语句返回 true❶，因为我们要让键盘留在屏幕上，让用户能够输入下一个猜测。return 语句告诉文本框，我们提供的事件处理代码是否结束了事件处理。通过返回 true，我们告诉 Android，为检查用户的猜测而需要做的事情都做了。如果返回 false，Android 将这样结束按键处理，即将数字键盘从屏幕上删除，就像你在网页表单中结束输入时那样。我们不希望键盘在用户输入猜测后消失，因此返回 true。

4.9.3　最后的润色

请运行这个应用多次，对其进行测试。你将发现按回车键可快速、高效地提交猜测；另外，输入的数字在文本框中居中。然而，当你获胜后，可能出现类似于图 4-31 所示的对齐问题，这取决于 API 版本以及屏幕的尺寸和像素密度。

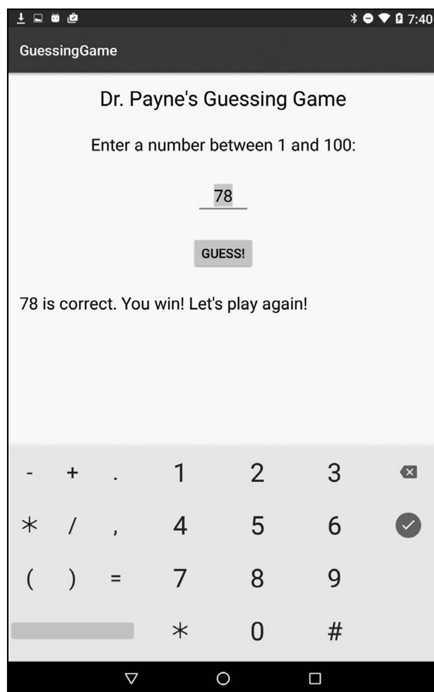


图 4-31　如果标签 lblOutput 中的文本没对齐，可能需要增大尺寸并让文本居中

这里的问题是，标签 lblOutput 会自动调整尺寸，以显示较长的文本 You win! Let's play again!，但有些较旧的 API 没有让这个标签居中。要避免这个问题，可在设计预览中将标签 lblOutput 增大到与屏幕等宽，并将其 textAlignment 属性改为 center。至此，这个猜数游戏就完成了，你可在自己的设备上玩，还可分享给朋友。图 4-32 显示了最终版本在 Nexus 7（字体设置为“大”）上的运行情况。

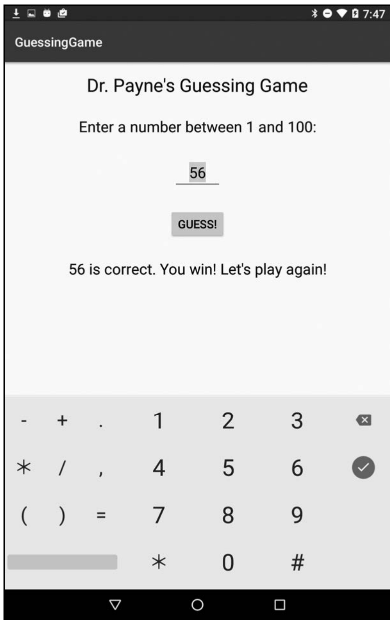


图 4-32 完成所有用户体验改善后的移动版猜数游戏

至此，你依次编写了一个基于文本的命令行猜数游戏、精致的 GUI 桌面版以及可在 Android 设备上运行的功能齐备的移动版。你感觉到 Java 的强大威力和灵活性了吗？第 5 章将添加设置菜单以及存储用户首选项的功能，让这个应用显得更专业！

4.10 小结

如你所见，在移动版猜数游戏中，可重用基于文本的版本和 GUI 桌面版本中的大量代码，这多亏了 Java 代码的跨平台兼容性。在本章中，你还学会了多项移动开发技能：

- 在 Android Studio 中新建项目；
- 在 Android Studio 设计视图中创建 GUI 布局，包括在 Properties 面板中修改各种界面元素的属性；
- 给布局中的 GUI 组件命名，以方便在 Java 代码中使用它们；
- 将 Android GUI 布局元素关联到 Java 代码；
- 在 Android 应用中添加方法，如 `checkGuess()` 和 `newGame()`；
- 在 Android 应用中重用 Java 代码；
- 在 Android 中处理事件，包括按钮单击事件以及键盘/编辑器操作（如按回车键）；
- 使用 Android 模拟器在 Android 虚拟设备上运行应用，以对其进行测试；
- 在 Android 手机、平板电脑和其他设备上启用开发者模式和 USB 调试，以便运行应用；
- 修改控件的属性并进行对用户友好的润色，以改善用户体验。

4.11　编程练习

为复习并使用学到的知识，以及获得更多的编程技能，请尝试完成这里的编程练习。如果遇到困难，可从 https://www.nostarch.com/learnjava/下载示例解决方案。

4.11.1　编程练习 1：指出用户猜了多少次

在第 3 章的编程练习中，你修改了获胜消息，以指出用户猜了多少次才猜对：

```
62 is correct! You win after 7 tries!
```

请使用 Toast 在 Android 版中也添加类似的功能。Toast 是一个 Android 控件，用于创建弹出式消息窗口，这是一种显示简单说明（如错误或指出用户赢了）的便利方式。图 4-33 所示的屏幕底部显示了一条 Toast 消息。

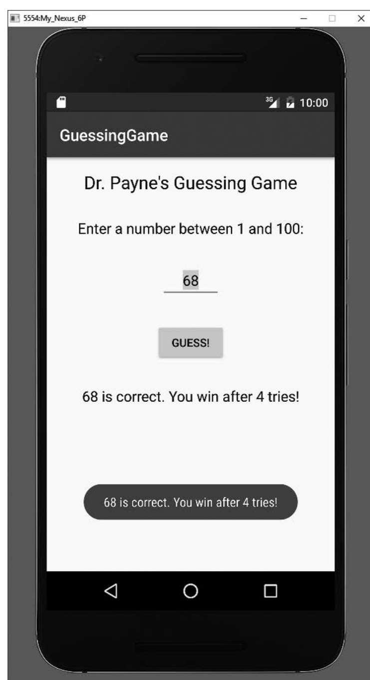


图 4-33　屏幕上的弹出式 Toast 消息窗口，向用户提供重要的信息

要创建弹出式 Toast 消息窗口，可使用方法 Toast.makeText()。你可根据下面的示例在指出用户获胜的 else 语句中显示一个 Toast 弹出窗口：

```
else {
    message = guess +
            " is correct. You win after " + numberOfTries + " tries!";
```

```
        Toast.makeText(MainActivity.this, message,
                Toast.LENGTH_LONG).show();
        newGame();
    }
```

上述代码在 MainActivity（猜数游戏）上弹出 Toast 窗口，在其中显示 String 变量 message 的内容，并持续几秒钟（因为指定了 Toast.LENGTH_LONG）。还有设置 Toast.LENGTH_SHORT，它导致消息弹出后马上消失，用户很难看清楚。末尾的方法.show()完成了一项重要工作——将 Toast 显示到屏幕上。

与 GUI 桌面版中一样，要完成这项任务，需要在类开头再创建一个变量（如 int numberOfTries = 0;），在每次调用方法 checkGuess()时都将猜测次数加 1，并在用户获胜时修改输出 message，以便在 Toast 弹出窗口和标签 lblOutput 中都显示猜测次数。添加计算猜测次数的功能后，再添加第 3 章介绍的用完指定的猜测次数就算输的功能：限定用户最多猜 7 次，并在用户每次猜测后都指出他还可猜多少次。

4.11.2 编程练习 2：提高视觉吸引力

在 Android Studio 中，探索 GUI 组件的一些美学属性，如背景色、前景色、字体、字号等。尝试移动猜数游戏的组件，让界面更具视觉冲击力。例如，可像图 4-34 那样定制这个游戏。

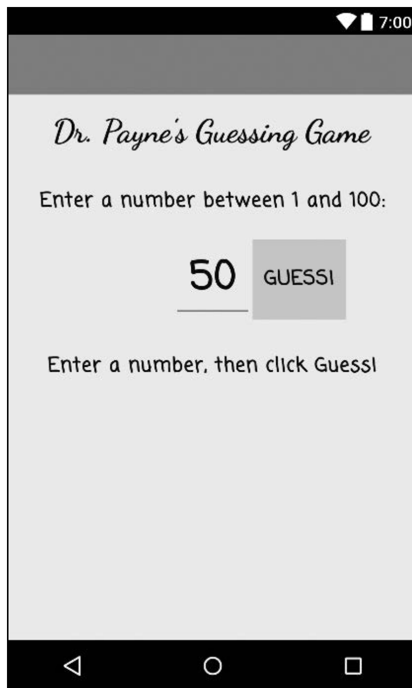


图 4-34 修改组件的颜色、字体、字号和排列，让应用引人注目！

你还可添加自定义背景图像（在文件夹 app▸res▸drawable 中添加一个图像文件，再将布局、按钮或其他组件的 background 属性设置为它）。你可尝试各种设置，并随时按 CTRL-Z 来撤销所做的操作。

4.11.3　编程练习 3：创建移动版 MadLibs 游戏

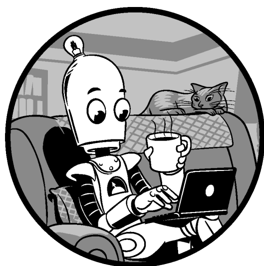
回过头去看看你在第 3 章为完成编程练习 3 创建的程序 MadLibGUI.java，再创建一个移动版 MadLibs 游戏，它显示一个图形用户界面，其中包含让用户输入多个单词的标签和文本框，如 txtBigAnimal、txtPastTenseVerb、txtSmallContainer 和 txtFood，还有一个用户可通过点击来生成 MadLibs 式故事的按钮。一个不错的主意是，在每个文本框都中包含一些默认或初始文本，让用户知道这个程序的工作原理。

用户单击按钮时，程序应在一个 TextView 或 EditText 控件中显示生成的 MadLibs 故事，但在 EditText 中显示更佳，因为这样用户可通过复制并粘贴来分享生成的故事。尝试使用不同的颜色、字体和布局，对结果满意后再与朋友分享这个程序。

提示　要让 TextView 或 EditText 显示多行文本，可使用转义字符序列\n 在使用方法 setText() 显示的字符串消息中插入换行符，如"Once upon a time... \n There was abuffalo princess \n wholived in a soup can."。这句引文将横跨 3 行——在转义序列\n 出现的每个地方都换行。

第 5 章

给应用添加菜单和首选项



你编写了一个有趣的 Android 应用，但它还缺少一些功能。你还没学习如何在 Android 中创建设置（选项）菜单，以及如何存储最高得分、游戏统计数据和其他信息。本章将给猜数游戏添加选项菜单以及存储信息的功能。

5.1　在 Android 中添加选项菜单

大多数应用和游戏都包含用户可通过菜单访问的选项（设置）。就猜数游戏而言，你可能想让用户能够修改难度、重新开始、查看统计数据或打开 About 屏幕。下面来创建一个让用户能够执行这些操作的菜单。

在 Android 中添加选项菜单的过程包含 4 个步骤。

(1) 编辑应用的默认 XML 菜单文件，创建可作为选项供用户选择的菜单项。

(2) 修改应用的活动文件，以显示第(1)步创建的菜单和选项。

(3) 创建在用户选择选项时做出响应的事件处理程序。

(4) 编写用户选择各个选项时将执行的代码。

添加选项菜单不仅让应用显得更专业，还让用户能够更好地控制游戏体验。我儿子很喜欢能够将猜测范围从 1~10 改为 1~100 乃至 1~1000 的功能，但添加显示获胜次数的 Game Stats 选项后，他们更是抱着设备不放，渴望让获胜次数越来越多。但愿你像我儿子一样，也觉得这些额外的功能很有趣乃至让人上瘾。

5.1.1　在 XML 菜单文件中添加菜单项

在 Android Studio 中打开猜数游戏项目，并在 Project Explorer 面板顶部将视图切换到 Android。然后，展开文件夹 app▸res▸menu，并双击 menu_main.xml 打开这个默认菜单文件。

将文件 menu_main.xml 的代码修改成下面这样：

```
<menu xmlns:android="http://schemas.android. com/apk/res/android">
    <item
        android:id="@+id/action_settings"
        android:title="Settings" />
    <item
        android:id="@+id/action_newgame"
        android:title="New Game" />
    <item
        android:id="@+id/action_gamestats"
        android:title="Game Stats" />
    <item
        android:id="@+id/action_about"
        android:title="About" />
</menu>
```

标签<menu>使用 Android XML 文档命名空间创建一个菜单资源，该命名空间是由统一资源标识符（URI）http://schemas.android.com/apk/res/android 标识的。可使用 XML 来存储或显示从网页到数据库等元素，为此只需将 XML 标签关联到它们。上述代码中的属性 xmlns（XML 命名空间）选择了主 Android 命名空间，因此这个 XML 文件中的标签表示 Android 应用中的常见元素。标签<menu>表示 Android 菜单，而每个<item>标签都描述了一个菜单项及其属性。这个菜单包含 4 个选项：Settings、New Game、Game Stats 和 About，因此我们添加了 4 个<item>标签。我们将这些名称赋给了菜单项的 title 属性，这个属性决定了用户打开菜单时，菜单项显示的文本。在本章后面，我们将使用属性 id 来确定用户选择了哪个菜单项。

将文件 menu_main.xml 存盘。现在该在猜数游戏中显示这个选项菜单了。

5.1.2　显示选项菜单

菜单已创建好，但要显示它，需要在文件 MainActivity.java 中添加一些 Java 代码。请在 Project Explorer 面板中，打开文件夹 app▸java▸com.*yourdomain*.GuessingGame 中的文件 MainActivity.java。

在 MainActivity 类的中间或末尾附近，有一个名为 onCreateOptionsMenu()的方法，将其代码修改成下面这样（如果你的代码中没有方法 onCreateOptionsMenu()，请在方法 onCreate()和 MainActivity 类的右大括号之间添加下述代码）：

```
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menu_main, menu);
        return true;
    }
} // 文件 MainActivity.java 末尾的右大括号
```

顾名思义，onCreateOptionsMenu()告诉 Android 如何为应用创建选项菜单。在这里，我们告诉 Android，我们要扩展文件 menu_main.xml，以用作选项菜单。文件 menu_main.xml 不是菜单，

因此需要使用 MenuInflator 类将其转换为菜单。我们使用方法 getMenuInflator()创建一个名为 inflater 的 MenuInflator 实例，再调用方法 inflate()并向它传递一个 XML 文件（R.menu.menu_main）以及要将该文件中的菜单项注入其中的菜单（menu）。在文件中添加这些代码时，你可能需要按 Alt-回车键（在 macOS 中是 Option-回车键）来添加缺失的 import 语句。

完成这些修改后，将文件存盘并运行应用。Android 将在应用的操作栏中显示 3 个句点，让你知道有一个选项菜单，如图 5-1（上）所示。单击这 3 个句点将显示选项菜单，如图 5-1（下）所示。

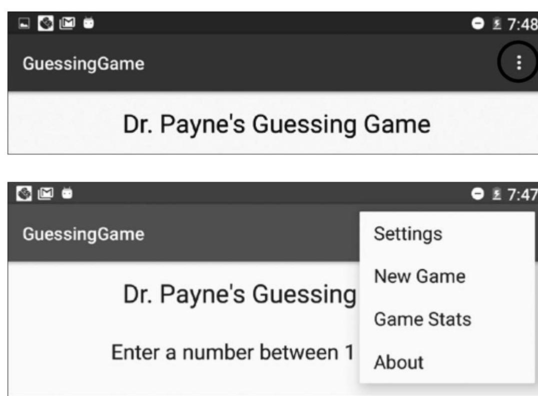


图 5-1 选项菜单显示为操作栏中的 3 个点（上）；单击这 3 个点将显示选项菜单（下）

如果你此时单击选项，什么都不会发生，因为还没有添加响应用户选择的代码。下面就来这样做。

5.1.3 响应用户选择

我们要让这个应用在用户选择菜单项时执行相应的操作，为此需要添加一个事件处理程序来跟踪用户选择的菜单项。我们将使用菜单项的 id 属性来确定用户的选择。

在文件 MainActivity.java 中，找到并修改事件处理程序 onOptionsItemSelected()；如果找不到这个方法，请在前一节修改的方法 onCreateOptionsMenu()和文件末尾的右大括号之间添加它。

```
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu_main, menu);
    return true;
}
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_settings:
            return true;
        case R.id.action_newgame:
            newGame();
            return true;
```

```
                case R.id.action_gamestats:
                    return true;
                case R.id.action_about:
                    return true;
                default:
                    return super.onOptionsItemSelected(item);
            }
        }
}
```

在这些代码中，我们使用了一条 switch 语句来确定用户选择的是哪个菜单项。switch 语句提供了另一种检查多个条件的途径，类似于一长串 if-else 语句。这里没有使用 4 条串接的 if-else 语句来检查可能选择的每个菜单项，而是使用了一条 switch 语句，并在关键字 switch 后面的括号内指定要检查的变量。在这个示例中，要检查用户选择的菜单项的 id，因此使用了 switch (item. getItemId())。接下来，在 switch 语句的大括号内，我们使用 case 语句列出了要检查的值（如 case R.id.action_settings）；在每条 case 语句后面都加上冒号（:），然后是运行的代码以及 break 或 return 语句。这个事件处理程序返回一个布尔值，因此我们在每个 case 块中都使用了 return（而不是 break）语句。如果这些地方没有 return 语句，就必须在每个 case 块的末尾添加 break 语句。

在这里，每条 case 语句都检查我们在文件 menu_main.xml 中输入的一个菜单项的 id 值，并执行相应的代码。当前，只有检查 action_newgame 值的 case 语句包含要执行的代码——调用方法 newGame()开始新游戏。其他 case 语句还需添加一些代码，下面依次来完成这些任务。

5.1.4　创建表示 About 屏幕的弹出式提醒框

用户选择菜单项 About 时，我们将像其他应用那样弹出一个对话框，为此我们将使用**提醒框**。提醒框是一种灵活的弹出框，用于向用户显示信息或提示用户做出响应。相比于第 4 章使用的 Toast 弹出框（参见该章的编程练习 1），这种弹出框的适应能力更强，因为 AlertDialog 类允许我们通过子类 Builder 来定制弹出框的属性。在这里，我们将在用户选择菜单项 About 时显示一个提醒框，其中包含一条消息，指出用户正在玩的杰出猜数游戏是谁开发的。

请在检查菜单项 action_about 的 case 语句中添加如下代码：

```
case R.id.action_about:
 ❶ AlertDialog aboutDialog = new AlertDialog.Builder(MainActivity.this).create();
 ❷ aboutDialog.setTitle("About Guessing Game");
 ❸ aboutDialog.setMessage("(c)2018 Your Name.");
 ❹ aboutDialog.setButton(AlertDialog.BUTTON_NEUTRAL, "OK",
            new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int which) {
                 ❺ dialog.dismiss();
                }
           });
 ❻ aboutDialog.show();
   return true;
```

我们使用 AlertDialog.Builder 类❶创建一个定制的弹出窗口。❷处的代码将这个弹出窗口的标题设置为 About Guessing Game，而❸处的代码行显示一条简单的消息，其中包含版权信息和你的姓名（但你可根据自己的喜好修改这些文本）。方法 setButton()❹在这个弹出窗口中添加一个 OK 按钮，而接下来的事件监听器 onClick()在用户单击 OK 按钮时，调用方法 dismiss()来关闭这个弹出窗口❺。最后，调用方法 show()显示这个定制的弹出窗口❻。

请按 Alt-回车键（或 Option-回车键）导入 AlertDialog 类，再保存所做的修改并运行应用。当你单击选项菜单并选择 About 时，将看到一个类似于图 5-2 的弹出窗口。

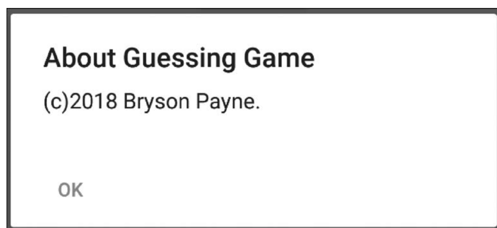


图 5-2 定制的弹出式提醒框

这个猜数游戏更像专业的 Android 应用了！接下来进入最精彩的部分，让用户能够选择游戏难度并记录用户赢了多少次。

5.2 修改猜测范围

让用户选择猜测范围（如从 1~10 改为 1~100 乃至 1~1000）是巨大的改进。明白了选项菜单和提醒框后，我们来筹划如何升级这款游戏，让用户能够修改猜测范围。

首先，需要添加一个表示范围的变量，这样可不使用硬编码值 100，而可使用用户选择的范围。其次，需要从两个方面修改这个应用的行为：修改方法 newGame()以使用新增的范围变量；让当前显示 Enter a number between 1 and 100:的 TextView 根据用户选择的范围显示不同的提示语。最后，需要给用户提供选择范围的途径；为此我们将创建一个定制的提醒框，其中包含 3 个可供选择的范围：1~10、1~100 和 1~1000。

5.2.1 添加表示范围的变量

首先，将计算随机数时使用的硬编码值 100 替换为一个变量。在 MainActivity 类的开头，声明变量 range 并将其设置为默认值 100：

```
public class MainActivity extends AppCompatActivity {
    private EditText txtGuess;
    private Button btnGuess;
    private TextView lblOutput;
    private int theNumber;
    private int range = 100;
```

既然在添加变量，我们顺便再添加一个 TextView 变量，用于表示当前显示 Enter a number between 1 and 100:的标签。用户选择除 1~100 外的其他范围时，这个标签将不再正确，因此我们需要通过一个变量来让这个标签显示合适的文本。我们将这个变量命名为 lblRange：

```
    private int range = 100;
    private TextView lblRange;
```

为将 GUI 关联到变量 lblRange，请在方法 onCreate()中添加如下代码行：

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    txtGuess = (EditText) findViewById(R.id.txtGuess);
    btnGuess = (Button) findViewById(R.id.btnGuess);
    lblOutput = (TextView) findViewById(R.id.lblOutput);
    lblRange = (TextView) findViewById(R.id.textView2);
```

如果出现错误，请检查设计视图中显示提示语的 TextView 的名称：展开文件夹 app▸res▸layout，打开其中的文件 content_main.xml，再单击内容为 Enter a number between 1 and 100:的标签，并将其 id 属性改为 textView2。

创建变量 range 和 lblRange 后，该修改应用的行为，让其使用这些变量而不是硬编码值了。

5.2.2　使用变量 range

首先，修改方法 newGame()，使其使用变量 range。还需添加必要的代码，以修改提示语让用户知道猜测范围：

```
public void newGame() {
    theNumber = (int)(Math.random() * range + 1);
    lblRange.setText("Enter a number between 1 and " + range + ".");
    txtGuess.setText("" + range/2);
    txtGuess.requestFocus();
    txtGuess.selectAll();
}
```

除使用变量 range 来生成相应范围内的随机数外，我们还根据变量 range 修改了 lblRange 显示的提示语。最后 3 行代码做了点润色工作：将文本框 txtGuess 中的默认值设置为变量 range 的一半。这样，如果猜测范围为 1~10，这个文本框显示的默认值将为 5；如果范围为 1~1000，默认值将为 500。

与范围相关的最后一项修改是方法 checkGuess()。为处理无效用户输入，前面添加了一条 try-catch 语句，并在 catch 语句中让用户输入位于范围 1~100 的整数。这里需要修改 catch 语句，以指出可供用户选择的范围：

```
public void checkGuess() {
--snip--
```

```
            }
        } catch (Exception e) {
            message = "Enter a whole number between 1 and " + range + ".";
        } finally {
--snip--
        }
    }
```

现在两个标签都将正确地显示用户指定的范围。现在该创建提醒框，让用户能够选择游戏难度了。

5.2.3 创建让用户选择范围的对话框

每当用户选择菜单项 Settings 时，都应显示包含各种范围选项（1~10、1~100 和 1~1000）的设置对话框。要显示这些选项，需要再创建一个定制的提醒框，它显示一个包含 3 个范围选项的列表视图。

首先，向后滚动到方法 onOptionsItemSelected()，并在 case R.id.action_settings 语句中添加如下代码：

```
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_settings:
            final CharSequence[] items = {"1 to 10", "1 to 100", "1 to 1000"};
            AlertDialog.Builder builder = new AlertDialog.Builder(this);
            builder.setTitle("Select the Range:");
            builder.setItems(items, null);
            AlertDialog alert = builder.create();
            alert.show();
            return true;
```

这 6 行代码显示一个提醒框，其中的列表视图包含 3 个猜测范围选项，但我们还需添加一些代码，以在用户选择时做出响应。方法 builder.setItems()除接受选项列表外，还可接受一个事件监听器，用于在用户选择列表项时做出响应。

如果用户选择第一个选项，我们就需要将变量 range 的值改为 10；同样，在用户选择第二或第三个选项时，我们需要将这个变量设置为 100 或 1000。builder.setItems()语句中的事件监听器代码如下所示：

```
        case R.id.action_settings:
            final CharSequence[] items = {"1 to 10", "1 to 100", "1 to 1000"};
            AlertDialog.Builder builder = new AlertDialog.Builder(this);
            builder.setTitle("Select the Range:");
            builder.setItems(items, new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int item) {
                    switch(item) {
                        case 0:
                            range = 10;
                            newGame();
```

```
                    break;
                case 1:
                    range = 100;
                    newGame();
                    break;
                case 2:
                    range = 1000;
                    newGame();
                    break;
            }
            dialog.dismiss();
        }
    });
    AlertDialog alert = builder.create();
    alert.show();
    return true;
```

请注意，用户选择列表项后，我们修改变量 range 的值，调用 newGame()生成指定范围内的随机数，并根据新指定的范围修改屏幕上显示的提示语。

请在完成这些修改后将文件存盘，并运行这个游戏以测试这些新选项：将范围修改为 1~-10 并玩几轮，再将范围改回到 1~100；如果你勇气可嘉，尝试将范围改为 1~1000。

关闭这个游戏后再运行它，你将发现它并没有记住你前一次选择的范围。这个游戏也没有记录你在猜数方面表现有多棒。要是能够让这个游戏记住你选择的范围以及你赢了多少次，那该有多好。

5.3　存储用户首选项和游戏统计信息

要记住用户首选项和游戏统计数据,关键在于将它们作为**持久化信息**保存到 Android 设备中。持久化信息指的是在应用关闭后依然保留在设备中的数据。在猜数游戏中，我们要将用户选择的难度以及获胜次数作为持久化信息存储起来。

要将持久化数据保存到 Android 设备，方式有三种：存储到共享首选项中；保存到文件中；保存到数据库中。**共享首选项**是一种对象，用于存储较短的设置清单，供应用下次启动时使用。之所以称为**共享**首选项，是因为你可在应用的多个活动或屏幕（如猜数游戏的选项菜单和主游戏屏幕）之间共享这些设置。在需要存储大量数据，如文本文档时，将文件保存到设备很有用；而对于地址簿或联系人清单等应用，数据库必不可少。在猜数游戏中，只需存储几个数字，因此我们将使用共享首选项。

5.3.1　存储和获取用户选择的范围

共享首选项被存储为一系列**键–值对**，其中每个值都有用于检索它的相关键。例如，可能有键–值对"range"和"100"，其中"range"是**键**，而"100"是存储在这个键下的**值**。下面来编写一个方法，将用户选择的范围存储到共享首选项中。

在文件 MainActivity.java 的末尾附近，在方法 onOptionsItemSelected()和最后的右大括号之间添加如下方法：

```
            default:
                return super.onOptionsItemSelected(item);
        }
    }
    public void storeRange(int newRange) {
        SharedPreferences preferences =
            PreferenceManager.getDefaultSharedPreferences(this);
        SharedPreferences.Editor editor = preferences.edit();
        editor.putInt("range", newRange);
        editor.apply();
    }
}
```

在你创建的每个应用中，都有一个默认的共享首选项对象，要访问它，可创建一个与它相关联的 SharedPreferences 变量。要获取默认的共享首选项对象，可对创建并维护共享首选项列表的对象 PreferenceManager 调用 getDefaultSharedPreferences()。别忘了在输入代码的同时导入必要的类，你也可在输入代码后按 Alt-回车键（或 Option-回车键）来导入。

要写入共享首选项，必须使用一个 Editor 对象，它让我们能够编辑各个共享首选项值。为存储特定的键-值对，我们使用一个 put 方法，如存储字符串值的 putString、存储整数的 putInt、存储浮点数的 putFloat、存储 true/false 的 putBoolean 等。每当用户选择新范围时，我们都将变量 range 作为参数 newRange 传递给方法 storeRange()。为将 newRange 存储在键"range"下，我们使用 editor.putInt("range", newRange);将用户指定的新范围值（10、100 或 1000）存储在共享首选项键"range"下。方法 apply()告诉 Android，你已结束对共享首选项值的修改，可以让修改生效了。

编写将范围存储到共享首选项中的方法 storeRange()后，需要在 onOptionsItemSelected()中的事件监听器中的每条 case 语句中调用这个方法：

```
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_settings:
            final CharSequence[] items = {"1 to 10", "1 to 100", "1 to 1000"};
            AlertDialog.Builder builder = new AlertDialog.Builder(this);
            builder.setTitle("Select the Range:");
            builder.setItems(items, new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int item) {
                    switch(item) {
                        case 0:
                            range = 10;
                            storeRange(10);
                            newGame();
                            break;
                        case 1:
                            range = 100;
```

```
                    storeRange(100);
                    newGame();
                    break;
                case 2:
                    range = 1000;
                    storeRange(1000);
                    newGame();
                    break;
            }
            dialog.dismiss();
        }
    });
    AlertDialog alert = builder.create();
    alert.show();
    return true;
```

最后，需要在游戏加载时检索存储的范围，这样游戏每次运行时，都将使用用户前一次选择的范围。请向上滚动到方法 onCreate()，并在其中添加下面两行从共享首选项中检索范围的代码：

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    txtGuess = (EditText) findViewById(R.id.txtGuess);
    btnGuess = (Button) findViewById(R.id.btnGuess);
    lblOutput = (TextView) findViewById(R.id.lblOutput);
    lblRange = (TextView) findViewById(R.id.textView2);
    SharedPreferences preferences =
        PreferenceManager.getDefaultSharedPreferences(this);
    range = preferences.getInt("range", 100);
    newGame();
```

请注意，我们在调用方法 newGame()前检索共享首选项，旨在确保应用重启后使用以前的范围。方法 getInt()查找存储在"range"键下的值，但如果没有找到值，就使用第二个参数指定的默认值（这里为 100）。我们这样做旨在确保用户首次运行应用时也有范围值。

请将文件存盘，并运行这个应用。这次选择不同的范围，再将应用关闭。你选择的范围将被保存，供你下次启动应用时使用。

5.3.2　存储获胜次数

高分、排行榜、连胜次数等记录成果的东西，通常都会激励玩家更努力、玩得更久并力图打破记录。对于这个游戏，我们要做的最后润色是添加记录获胜次数的功能。同样，我们可轻松地将这些统计数据存储到共享首选项中。

用户猜对数字从而赢得一轮后，我们可使用共享首选项来检索其获胜次数、将这个数字加 1 并存储结果。为此，请在方法 checkGuess()中添加如下代码——放在处理用户猜对的 else 语句中：

```
public void checkGuess() {
    String guessText = txtGuess.getText().toString();
```

```
String message = "";
try {
    int guess = Integer.parseInt(guessText);
    if (guess < theNumber)
        message = guess + " is too low. Try again.";
    else if (guess > theNumber)
        message = guess + " is too high. Try again.";
    else {
        message = guess +
                " is correct. You win! Let's play again!";
      ❶ SharedPreferences preferences =
            PreferenceManager.getDefaultSharedPreferences(this);
      ❷ int gamesWon = preferences.getInt("gamesWon", 0) + 1;
      ❸ SharedPreferences.Editor editor = preferences.edit();
      ❹ editor.putInt("gamesWon", gamesWon);
      ❺ editor.apply();
        newGame();
    }
```

在这里，我们访问默认的 SharedPreferences 对象❶，检索存储在"gamesWon"键下的值❷（用户首次获胜时，这个值默认为 0），再将其加 1。在❸处，创建一个编辑器，用于将新值写入共享首选项。在❹处，将整数值 gamesWon 放到共享首选项的"gamesWon"键下。在❺处，我们告诉 Android 将修改写入到设备中。

这些代码存储获胜次数，但如何向用户显示这些统计数据呢？为此，需要在方法 onOptionsItemSelected()中的 case R.id.action_gamestats 语句中添加代码，如下所示：

```
case R.id.action_gamestats:
❶ SharedPreferences preferences =
      PreferenceManager.getDefaultSharedPreferences(this);
❷ int gamesWon = preferences.getInt("gamesWon", 0);
❸ AlertDialog statDialog = new AlertDialog.Builder(MainActivity.this).create();
  statDialog.setTitle("Guessing Game Stats");
❹ statDialog.setMessage("You have won "+gamesWon+" games. Way to go!");
  statDialog.setButton(AlertDialog.BUTTON_NEUTRAL, "OK",
          new DialogInterface.OnClickListener() {
              public void onClick(DialogInterface dialog, int which) {
                  dialog.dismiss();
              }
          });
  statDialog.show();
  return true;
```

在❶处，我们关联到应用的默认共享首选项；在❷处，检索获胜次数（如果这是用户首次运行该应用，我们将使用默认值 0）。在❸处，创建一个提醒框，用于显示用户赢了多少次；在❹处，显示获胜次数和一条鼓励消息。

请保存所做的修改并运行这个游戏，你可能根本停不下来！图 5-3 显示了获胜很多次后出现的 Game Stats 屏幕。

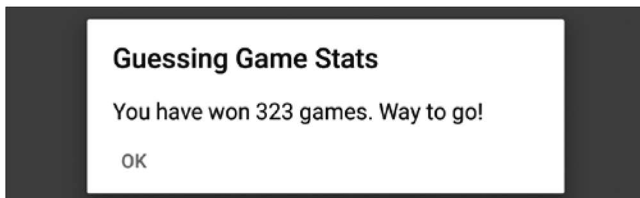


图 5-3　Game Stats 屏幕显示玩家猜对了多少次

添加选项菜单、保存游戏统计数据和用户首选项、显示提醒框，这些都是可对游戏或其他应用做的润色，让它显得既专业又功能齐备。请不断改进你的应用，添加想到的新功能，最终你将得到一个值得与朋友乃至所有人分享的应用。祝你编程愉快！

5.4　小结

通过对猜数游戏做几方面的润色，你创建了一个专业品质的 Android 移动游戏。这些润色包括：

- 给 Android 应用添加选项菜单；
- 通过编辑 XML 菜单文件来设计选项菜单；
- 使用 `MenuInflater` 显示选项菜单；
- 在用户选择菜单项时做出响应；
- 用包含多个 `case` 的 `switch` 语句替代冗长的 `if-else` 链；
- 使用 `AlertDialog` 类创建定制的弹出窗口；
- 使用 `SharedPreferences` 类存储共享首选项和应用统计数据；
- 在应用启动时检索用户的共享首选项。

5.5　编程练习

为复习并使用学到的知识以及获得更多的编程技能，请尝试完成这里的编程练习。如果遇到困难，可从本书配套网站（https://www.nostarch.com/learnjava/）下载示例解决方案。。

5.5.1　编程练习 1：有赢有输

第 4 章的编程练习 1 让你要求用户在 7 次之内猜对一个 1~100 的数字。本章添加了修改范围的功能，因此你需要相应地修改最大猜测次数。

第 4 章说过，可使用二分查找策略来猜测 1~100 的数字（每次都猜可能范围内的中间值）。2^7（2 的 7 次方）为 128，这意味着如果使用二分查找法，最多只要 7 次就能猜对 1~128 内的数字。但要猜对 1~10 或 1~1000 内的数字，最多需要多少次呢？

要计算最多需要猜测多少次，需要知道至少要将 2 自乘多少次才能超过指定的范围。例如，

对于 1~10 的数字，由于 $2^4 = 16$，而 $16 > 10$，因此最多只要 4 次就能猜对；对于 1~1000 的数字，由于 $2^{10} = 1024$，因此最多需要猜 10 次。

要确定需要将一个数字自乘多少次才能大于另一个数字，可使用**对数**。对数将一个数字和一个底作为参数，并计算这个底的多少次方等于指定的数字。Java 提供了方法 `Math.log()`，它将一个数字作为参数，并计算 10 的多少次方等于这个数字。将两个数字以 10 为底的对数相除时，结果与以第二个数字为底的第一个数字的对数相同。这意味着通过将 `Math.log(range)`和 `Math.log(2)`相除，可知道 2 的多少次方为 `range`。由于计算得到的幂次可能为小数，而你给用户指定猜测次数时不想使用小数，如 7.25，因此需要加 1 并将结果转换为 `int`。要计算每种范围对应的最多猜测次数，可使用表达式`(int)(Math.log(range)/Math.log(2) +1)`。

请修改猜数游戏，使其根据用户选择的范围调整最大猜测次数——在游戏启动以及用户使用选项菜单选择新范围时都这样做。例如，你可创建一个名为 `maxTries` 的变量，并在检查用户是否用完给定的猜测次数时，用这个变量替换硬编码数字 `7`。

5.5.2　编程练习 2：胜率

完成编程练习 1 后，对猜数游戏进行修改，以存储获胜次数和失败次数。修改响应用户选择菜单项 Game Stats 的代码，以获取这两个数字并显示获胜次数、玩游戏的总次数以及胜率（获胜次数除以总次数，再乘以 100），如图 5-4 所示。

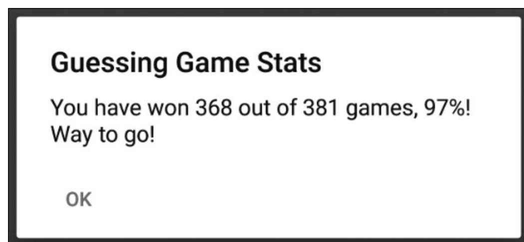
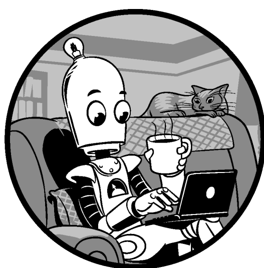


图 5-4　显示胜率的 Game Stats 屏幕

第 6 章

机密信息加密



你可能给朋友递过纸条，这些纸条是经过加密的，让父母或老师看不懂。本章将做类似的事情——创建应用 Secret Messages。

本书的前一个应用（猜数游戏）专注于数字——猜大了、猜小了或猜对了。相反，这个 Secret Messages 应用将专注于文本。你将学习如何使用 Java 来处理文本字符串，以及如何操作字符值来生成密信。

6.1 凯撒加密法

本章的应用将使用**凯撒加密法**对机密信息进行加密和解密。凯撒加密法是 2000 多年前发明的一种算法，通过替换字母来加密信息。

凯撒加密法因罗马大帝尤利乌斯 · 凯撒而得名（公元前 100—公元前 44 年）。据历史学家说，凯撒喜欢对私信（如给将军的手谕）进行加密，方法是对字母表中的字符进行“移位”。例如，图 6-1 所示的密码盘将字母移动 13 个位置。

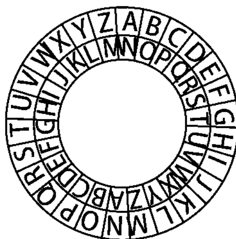


图 6-1　密钥为 13 的凯撒加密盘

在这个密码盘中，外面的字母对应于将替换它的内部字母，因此 A 将变为 N，B 将变为 O，以此类推。下面是一个使用这种加密法的示例：

```
Secret messages are so cool!
Frperg zrffntrf ner fb pbby!
```

第一行是**明文**——能够读懂的原始信息；第二行是**密文**——加密后的版本。

要将密文解密成明文，只需执行反向替换：对于每个字母，都在内圈找到它，再将其替换为对应的外圈字母。

并非所有的凯撒加密法都像这个一样是对称的。所谓**对称**，指的是加密和解密的方法相同。例如，加密时加 13 将 F 变成 S，解密时减 13 将 S 变成 F；换而言之，加密和解密使用的密码盘和**密钥**（key，字母的移动量）相同。字母移动量（这里是 13）之所以被称为密钥，是因为知道它就能破解（unlock）密码。我们将创建的应用 Secret Messages 支持使用任何密钥。

6.2 创建应用 Secret Messages

我们将像创建猜数游戏那样创建应用 Secret Messages：首先创建命令行版，然后创建桌面 GUI 版，最后创建 Android 移动版。命令行版看起来非常简单，如图 6-2 所示，但它让我们能够快速而轻松地测试加密算法。

```
Select Command Prompt

C:\Users\Bryson Payne\workspace_ch05\SecretMessages\bin>java SecretMessages
Enter a message to encode or decode:
Secret messages are so cool!
Enter a secret key (-25 to 25):
13
Frperg zrffntrf ner fb pbby!

C:\Users\Bryson Payne\workspace_ch05\SecretMessages\bin>_
```

图 6-2 命令行版 Secret Messages 应用

从图 6-2 可知，这个程序让用户输入要加密或解密的消息以及密钥值，然后显示加密或解密后的消息。请注意，刚开始编写这个程序时，将对整条消息（包括空格和标点）进行加密；但在最终的版本中，我们将添加一些逻辑，只对字母进行加密，如图 6-2 所示。

6.2.1 在 Eclipse 中创建项目 Secret Messages

首先启动 Eclipse。如果打开了以前的项目中的文件，请关闭它们。为新建 Java 项目，选择 File▸New▸Java Project，并将项目命名为 SecretMessages，再单击 Finish 按钮。

在 Package Explorer 面板中，展开项目文件夹 SecretMessages，右击其中的文件夹 src 并选择 New▸Class 以新建一个 Java 源代码文件,并将其也命名为 SecretMessages。选中复选框 public static void main(String[] args)以创建 `main()`方法，如图 6-3 所示。



图 6-3　新建一个 Java 项目，在其中新建一个名为 SecretMessages 的类文件，并选择指定要创建 main()方法存根的复选框

单击 Finish 按钮，你将在 Eclipse 主窗口中看到文件 SecretMessages.java。下面来着手给 Secret Messages 应用编写代码。

6.2.2　开始在 SecretMessages.java 中编写代码

在文件 SecretMessages.java 的开头（声明 public class SecretMessages 上方），添加导入 java.util.Scanner 的语句，以便能够让用户输入：

```
import java.util.Scanner;
public class SecretMessages {
    public static void main(String[] args) {
     ❶ Scanner scan = new Scanner(System.in);
     ❷ System.out.println("Enter a message to encode or decode:");
    }
}
```

在 main()方法中，我们创建了一个名为 scan 的 Scanner 对象❶，然后提示用户输入一条要加密或解密的消息❷。

提示用户输入后，我们将创建一个名为 message 的 String 变量，用于存储用户输入的文本行：

```
        System.out.println("Enter a message to encode or decode:");
        String message = scan.nextLine();
```

接下来，使用对象 scan 的方法 nextLine()获取用户输入的整行文本（到回车键为止），并将其存储在字符串变量 message 中。

至此，这个应用能够让用户输入一条消息，并将其存储到一个变量中。接下来，我们需要学习如何操作字符串中的字符，以创建消息的加密版本。请将文件存盘，再接着阅读下一节。

6.2.3 打乱字符串

到目前为止，应用 Secret Messages 与猜数游戏很像。我们创建一个输入扫描器，提示用户提供输入，通过扫描控制台获取用户提供的输入，并将其存储到一个变量中。应用 Secret Messages 与猜数游戏的不同之处是，它能够处理字符串中的字符。

要完成命令行版 Secret Messages 应用，需要很多步，因此我们将以**迭代**的方式创建它。这意味着不是一次性编写出整个应用，并期望最终的代码管用，而是一点点创建该应用（即逐步迭代）并边写边测试，确保应用在任何时点都能运行。每次迭代也许未提供所需的全部功能，但终将编写出功能齐备的完整应用。

我们需要使用文本处理来将输入的字符串转换为输出字符串。为在 Java 中处理文本字符串，我们将创建一个简单的消息反转器。换而言之，我们将接受用户输入的消息，再将其中的字母按相反的顺序排列。例如，对于字符串 Meet me at the arcade at 5pm，我们将把它变成 mp5 ta edacra eht ta em teeM。

首先，创建一个名为 output 的变量（用于存储反转后的字符串），并将其设置为空字符串：

6

```
String message = scan.nextLine();
String output = "";
```

我们需要一个循环来遍历消息中的字符。为方便起见，可使用一个 for 循环，因为我们知道消息包含的字符数（后面将使用方法 message.length()来确定消息包含多少个字符）。在 Java 中，要声明一个 for 循环，需要做三件事情：**初始化**一个循环变量；检查一个**条件**；进入下一次迭代前**更新**循环变量。我们使用分号将这三部分分开。for 循环声明语法类似于下面这样：

```
for ( initialization; condition; update ) { body }
```

例如，你可以打开 JShell 并输入如下代码来创建一个 for 循环：

```
jshell> for ( int x = 0; x < 10; x++ ) { System.out.println(x); }
```

这个循环打印数字 0~9。其中，初始化部分将循环变量 x 设置为 0；条件检查 x 是否小于 10；更新部分在每次迭代后都将 x 的值加 1。x++使用了被称为**递增运算符**的快捷方式，它在每次迭代后都将 x 递增（即加 1），这与语句 x = x + 1 等价。这个 for 循环在每次迭代中都打印 x 的值，这种操作重复 10 次，直到 x 不再小于 10。

为了反转消息字符串中字符的排列顺序，可将一个变量初始化为该字符串中最后一个字符的位置（**索引**），再反向遍历字符串中的字符（从最后一个到第一个）。在我们的示例中，这类似于

下面这样：

```
for ( int x = message.length()-1; x >= 0; x-- ) { }
```

在 Java 中，字符串中字符的位置编号为 0（第一个字符的索引）到字符串长度减 1，即第一个字符的索引为 0，第 *n* 个字符的索引为(*n* – 1)。

图 6-4 中显示了前述示例消息的前 10 个字符，其中每个字符的下方都显示了其索引。第一个字母（M）的索引为 0，第一个字母 e 的索引为 1，以此类推。请注意，空格也被视为字符。这里有两个空格，它们的索引分别为 4 和 7。在这条消息中，第 10 个字符为字母 t，其索引为 9。这种模式将不断重复，直到最后一个字符，其索引为 message.length()-1。

M	e	e	t		m	e		a	t
0	1	2	3	4	5	6	7	8	9

图 6-4　一条消息中的字符及其索引

我们要从字符串末尾开始创建反转消息，因此将 x 初始化为 message.length()-1——消息中最后一个字符的索引。条件为 x >= 0，因为我们要不断后移，直到到达第一个字符（其索引为 0）。最后，更新部分为 x--，因为我们要以每次一个字符的方式往后移。与 x++相反，x--使用了**递减运算符**——每次将 x 的值减 1。请回到 Eclipse，并在前面编写的最后一行代码后面开始编写这个 for 循环：

```
String output = "";
for ( int x = message.length()-1; x >= 0; x-- ) {
}
```

为获取字符串中特定位置的字符，我们调用方法 charAt()，并将要获取的字符的索引传递给它。为在字符串末尾附加（添加）字符，我们使用运算符+。将这些整合起来，就能创建字符串消息的反转版，并将其存储在字符串变量 output 中：

```
for ( int x = message.length()-1; x >= 0; x-- ) {
    output += message.charAt(x);
}
```

这个 for 循环的循环体只有一行代码：获取 message 中索引 x 处的字符，并将其添加到 output 中。别忘了将这个 for 循环的循环体放在大括号内，因为当这个应用增大时，我们需要在这个循环中添加代码行。

现在只需将消息 output 显示到屏幕上就可以了，为此可使用 System.out.println(output)。代码清单 6-1 显示了这个应用的“消息反转”迭代的完整代码。

代码清单 6-1　应用 Secret Messages 的第一次迭代：将用户输入的消息反转

```
import java.util.Scanner;
public class SecretMessages {
```

```
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter a message to encode or decode:");
        String message = scan.nextLine();
        String output = "";
        for ( int x = message.length()-1; x >= 0; x-- ) {
            output += message.charAt(x);
        }
        System.out.println(output);
    }
}
```

你可在 Eclipse 中运行这个程序，并使用自己的消息进行测试，如图 6-5 所示。这并非最终的 Secret Messages 应用，但它确实让消息更难看懂——虽然很容易解密。

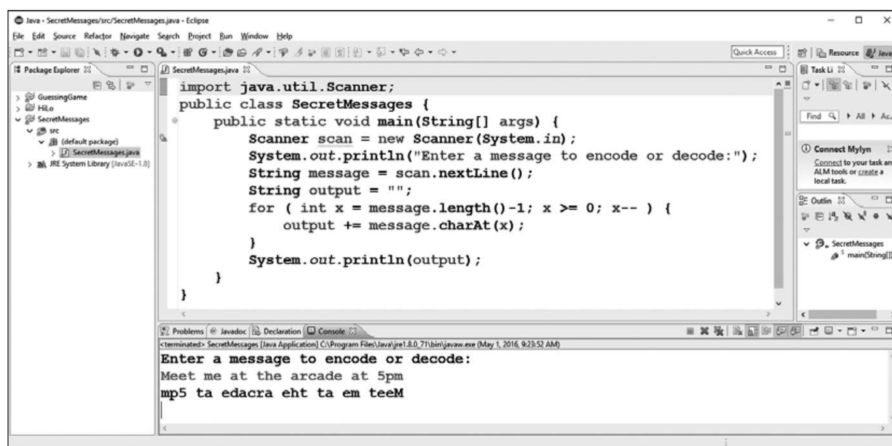


图 6-5 运行这个程序，并在屏幕底部的控制台窗口中输入要反转的消息

要对反转的加密消息进行解密，可复制它并作为输入提供给程序，如图 6-6 所示。你可使用这个程序对消息进行加密，并将加密后的消息发送给朋友，而朋友可使用这个程序将收到的消息解密。换而言之，你创建了第一个机密信息加密器！

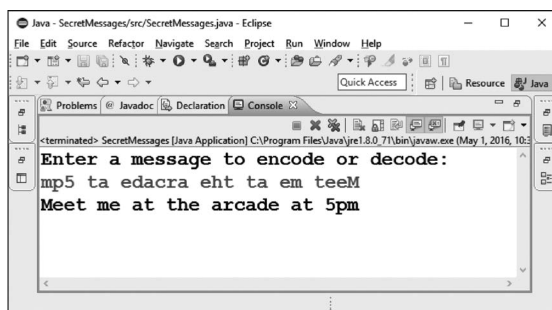


图 6-6 将加密后的消息作为输入提供给这里的程序，以将其解密

这个程序很简单，你无需使用它就能将加密后的消息解密，因此使用它来将要发送给朋友的消息进行加密并不安全。然而，你确实学会了如何使用 for 循环来遍历文本字符串、如何使用 length()来确定字符串的长度、如何使用 charAt()来访问位于字符串中特定位置（索引）处的字符，以及如何使用运算符+在字符串末尾添加字符。在下一节，你将学习如何修改字符串中字符的值，让字符串难以看懂，但对程序来说依然很容易解密。

6.3　Java 中的字符和值

要创建更出色的机密信息加密器，需要处理文本字符串中的字符值。要实现凯撒加密法，需要调整这些值，如将 A 改为 N 并将 N 改为 A。为此，需要明白字符是如何在计算机中存储的。

在 Java 中，可使用数据类型 char 来存储单个字符，如'A'。请注意，我们使用单引号来将字符值括起。字符串方法 charAt()返回表示单个字符的 char 值。Java 使用的是 16 位的 Unicode 字符。Unicode 是一个国际字符集，包含来自全球各地的数千个字符和符号，这些字符和符号是用数字值表示的。数据类型 char 是一种以数值方式存储 Unicode 字符（如'A'、'ñ'和'ç'）的方式。

与 int 变量一样，可将 Unicode char 值与整数相加。在应用 Secret Messages 中，我们要将'A'的 char 值（65）与密钥（13）相加，以获得新的 char 值（78）——它表示加密后的字母（'N'）。

在这次应用迭代中，将实现凯撒加密法。首先创建一个名为 key 的变量，并将其值设置为 13。为此，请在声明 String 变量 output 的代码行后面添加如下代码：

```
String output = "";
char key = 13;
```

还需修改 for 循环，使其从字符串开头遍历到末尾。这次我们从索引 0 开始遍历字符串，直到 x 不小于 message.length()。在每次循环中，我们都将表示字符位置的 x 加 1。修改后的 for 循环如下：

```
char key = 13;
for ( int x = 0; x < message.length(); x++ ) {
```

最后，不能将字符串变量 message 中的字符直接添加到字符串变量 output 中，而需要将每个字符与密钥值相加，得到可添加到字符串变量 output 中的 char 值：

```
    output += (char)(message.charAt(x) + key);
```

我们将字符串变量 message 中的每个字符都加上密钥值，再将结果转换为 char。通过在表达式前面加上用括号括起的 char，可让表达式的值可存储到数据类型 char 中，即将这个值**转换**为数据类型 char。这里为什么要将计算得到的结果转换为 char 呢？因为在 Java 中，这个表达式的结果默认为 int 值。代码清单 6-2 显示了完成这些修改后的程序。

代码清单 6-2 应用 Secret Messages 依然不长——只有 14 行代码，但现在能够加密文本字符串了

```
import java.util.Scanner;
public class SecretMessages {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter a message to encode or decode:");
        String message = scan.nextLine();
        String output = "";
        char key = 13;
        for ( int x = 0; x < message.length(); x++ ) {
            output += (char)(message.charAt(x) + key);
        }
        System.out.println(output);
    }
}
```

如果你现在运行这个程序，并输入一条消息，将看到类似于下面的输出：

```
Enter a message to encode or decode:
Secret messages are so cool!
`rp□r?-zr??ntr?-n□r-?|-p||y.
```

消息被加密，但与我们的期望不完全一致。首先，对包括空格和标点在内的所有字符都进行了加密；其次，没有从字母表末尾回绕到开头，因为我们将每个字符都加 13，而没有考虑到字母表末尾的字母需要回绕到字母表开头，这导致输出消息中包含怪异的符号和不可打印的字符。下一节将解决这两个问题，但在此之前，请将程序存盘。

6

6.4 只加密字母

Secret Messages 应用的第二次迭代能够提供加密后的消息，但缺乏一些必要的功能。为创建最终的版本，需要以 `if` 语句和条件的方式添加一些逻辑，以便只加密字母（而不加密空格和标点），同时在加密时让字母表从末尾回绕到开头。我们将在第三次迭代中完成这些改进。

首先，对输入消息中的每个字符进行检查，而不是直接将其添加到输出消息中。我们来修改 `for` 循环的循环体：

```
        for ( int x = 0; x < message.length(); x++ ) {
            char input = message.charAt(x);
        }
```

存储在变量 `input` 中的字符依次为 `message` 中的第一个字符、第二个字符等。我们需要逐个地检查字符，看看是否需要对其进行加密。我们要对字母进行加密：将其加上密钥值，并在必要时回绕到字母表开头。如果字符不是字母，而是空格或标点，就保持原样并将其添加到输出消息中。

请在 for 循环的循环体内添加如下 if 语句：

```
char input = message.charAt(x);
if (input >= 'A' && input <= 'Z')
```

在这条 if 语句的条件中，可使用 65（'A'的值），也可使用字符字面量'A'。这个条件检查存储在 input 中的字符是否大于等于'A'且小于等于'Z'，即 input 包含的是否是大写字母。如果条件满足，我们要给 input 加上密钥值，通过移动来生成用于替换原字母的凯撒加密值。请注意，这里只处理大写字母——小写字母需单独处理。

现在，在 if 语句的语句体内，我们要给 input 加上密钥值，对字母进行加密。我们还需在这里处理回绕问题：如果加上密钥值后超过了'Z'，就回绕到字母表开头。因此，for 循环内的代码将变成下面这样：

```
for ( int x = 0; x < message.length(); x++ ) {
    char input = message.charAt(x);
    if (input >= 'A' && input <= 'Z')
    {
     ❶ input += key;
     ❷ if (input > 'Z')
         ❸ input -= 26;
    }
 ❹ output += input;
}
```

从消息中获取下一个字符，并确定它是大写字母后，通过加上密钥值对其进行加密❶。接下来，检查加上密钥值后是否超过了 Z❷；如果是这样的，就从加密后的 input 值中减去 26（英语字母表包含的字母个数），以回绕到字母表开头❸。最后，将得到的 input 值添加到字符串变量 output 中❹。

如果你现在运行这个程序，将能够对全大写消息进行加密，如下所示：

```
Enter a message to encode or decode:
SECRET MESSAGES ARE SO COOL!
FRPERG ZRFFNTRF NER FB PBBY!
```

请注意，标点和空格保持不变，但所有大写字母都向后移 13 个位置并进行回绕，因此 S 将变成 F。

对小写字母进行加密的逻辑与大写字母相同。你可复制并粘贴前面的 if 语句代码，并将 A 和 Z 改为小写，但别忘了在第二条 if 语句前面添加 else。你可自己尝试完成这项任务，但如果遇到麻烦，可参阅下一节的代码清单 6-3 列出的完整代码。

加密使用其他语言编写的消息

这里编写的程序版本只能处理基本拉丁字符 A-Z 和 a-z。如果使用的语言不是英语，可调整这个程序，条件是你使用的语言包含的所有字母和符号在 Unicode 字符集中都连在一起。你可像本章的英语加密/解密程序那样，检查你使用的语言中的第一个和最后一个字符。如果你使用的语言包含的字符不是连在一起的，或者它同时使用基本拉丁字母和一些其他的字符（例如，西班牙语和法语在分别使用 ñ 和 ç 的同时使用其他重音字符），你可像代码清单 6-2 那样给所有字符都加上密钥值；也可保留空格不变（要检测空格，可使用方法 `Character.isSpace()`），并对其他字符都进行加密。请对你选择的语言尝试使用不同的加密方法，以找出最适合的加密方案。修改程序，使其提供一些新功能，这是最佳的学习方式！

6.5 关闭 Scanner 对象

这里可根据第 2 章的介绍做一点润色。`Scanner` 变量 `scan` 下面应该有条黄线，这是 Eclipse 发出的警告，指出存在资源泄露。如果你将鼠标指向这个警告，将出现消息`'scan' is never closed`。别忘了，使用完任何输入/输出资源（如 `Scanner` 对象）后，都必须将其关闭。为此，我们在方法 `main()`中最后一个 `System.out.println` 后面添加命令 `scan.close()`：

6

```
        System.out.println(output);
           scan.close();
```

添加这行代码后，资源泄露警告将消失。代码清单 6-3 列出了使用密钥值 `13` 的凯撒加密/解密程序的完整代码。

代码清单 6-3　这个应用版本是功能齐备的凯撒加密/解密程序，使用的密钥值为 13

```
import java.util.Scanner;
public class SecretMessages {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter a message to encode or decode:");
        String message = scan.nextLine();
        String output = "";
        char key = 13;
        for ( int x = 0; x < message.length(); x++ ) {
            char input = message.charAt(x);
            if (input >= 'A' && input <= 'Z')
            {
                input += key;
                if (input > 'Z')
                    input -= 26;
```

```
            }
            else if (input >= 'a' && input <= 'z')
            {
                input += key;
                if (input > 'z')
                    input -= 26;
            }
            output += input;
        }
        System.out.println(output);
        scan.close();
    }
}
```

请尝试运行这个程序并解密一条消息。然后，复制加密后的输出，再次运行这个程序，并在提示时粘贴加密后的消息并按回车键。程序将显示加密前的原始消息。下面显示了两次运行这个程序的情况——在第一次运行程序时复制加密得到的消息，并将其作为第二次运行时的输入：

```
Enter a message to encode or decode:
Secret messages are so cool!
Frperg zrffntrf ner fb pbby!
Enter a message to encode or decode:
Frperg zrffntrf ner fb pbby!
Secret messages are so cool!
```

由于这个程序使用的凯撒加密法是对称的，因此使用它来处理密文将解密得到最初的明文。这意味着你可使用代码清单 6-3 所示的程序加密消息，再将密文发给朋友，而朋友可使用这个程序进行解密。

遗憾的是，这意味着只要运行这个程序（或者说破解其解密算法），任何人都能够对密文进行解密。在下一次迭代中，我们将让这个程序更有趣些——允许用户指定要使用的密钥值；这样，对于要发送给不同的人的消息，可使用不同的密钥进行加密，也可在每次加密时都使用不同的密钥。

6.6　支持自定义密钥值

当前，这个使用密钥值 13 的机密信息加密程序效果很好，但如果要使用不同的密钥进行加密，如 3（经典的凯撒加密法）、5 或 25 呢？

除提示用户输入消息外，还需提示用户输入密钥值。为此，可在创建 char 变量 key 的代码行前面再显示一条提示语：

```
        String output = "";
        System.out.println("Enter a secret key (-25 to 25):");
        char key = 13;
```

为方便解密，我们将允许密钥值为负。如果你使用密钥值 5 对消息进行加密，再将其发送给

朋友，朋友可使用密钥值-5 进行解密。

用户输入所需的密钥值时，我们可扫描他提供的输入行，从中提取整数值并将其存储到一个 int 变量中：

```
String output = "";
System.out.println("Enter a secret key (-25 to 25):");
int keyVal = Integer.parseInt(scan.nextLine());
```

最后，不将变量 key 设置为 13，而是像下面这样设置它：

```
int keyVal = Integer.parseInt(scan.nextLine());
char key = (char) keyVal;
for ( int x = 0; x < message.length(); x++ ) {
```

在这里，我们将 keyVal 强制转换为 char，并将结果存储到变量 key 中，因为不能直接将整数存储到 char 变量中。

为使用负的密钥值进行解密，需要修改 if 语句中的逻辑：检查**减去**一个值（或者说加上一个负的密钥值）后，是否会越过字母表开头（即小于'A'）。如果是这样的，就再**加上** 26，使其位于范围 A-Z 内。请在处理大写字母的 if 语句中添加如下代码：

```
if (input >= 'A' && input <= 'Z')
{
    input += key;
    if (input > 'Z')
        input -= 26;
    if (input < 'A')
        input += 26;
}
```

如果指定的密钥为负值，就检查是否移到了 A 前面；如果是这样，就加上 26 以回绕到字母表末尾。

别忘了，对 else-if 语句中处理小写字母的逻辑做同样的修改，如下所示：

```
else if (input >= 'a' && input <= 'z')
{
    input += key;
    if (input > 'z')
        input -= 26;
    if (input < 'a')
        input += 26;
}
```

完成这些修改后，就可运行这个程序，并使用自定义密钥进行加密和解密了。程序的运行情况类似于下面这样：

```
Enter a message to encode or decode:
You've written a really cool app in Java!
```

6

```
Enter a secret key (-25 to 25):
7
Fvb'cl dypaalu h ylhssf jvvs hww pu Qhch!
Enter a message to encode or decode:
Fvb'cl dypaalu h ylhssf jvvs hww pu Qhch!
Enter a secret key (-25 to 25):
-7
You've written a really cool app in Java!
```

第一次运行这个程序时，我们使用了密钥值 7。为对密文进行解密，我们再次运行这个程序，并使用密钥值-7，结果为原来的明文。

大胆去尝试吧！

6.7　加密数字

前面对字母进行了加密，但如果我们对类似于下面的消息进行加密，数字将保持不变，依然处于明文状态：

```
Enter a message to encode or decode:
Meet me at the arcade at 5pm.
Enter a secret key (-25 to 25):
8
Ummb um ib bpm izkilm ib 5xu.
```

注意，在输出消息中，5pm 中的 5 还是 5。要对数字和字母都进行加密，需要在 for 循环中添加相应的逻辑。我们需要检查字符是否在范围 0~9 内，如果是，就将其加密为别的数字；此外还需回绕到数字集合 0~9 的开头或末尾。由于处理的是数字，因此回绕方式与字母表不同。

首先，在处理小写字母的 if-else 语句后面再添加一条 else-if 语句：

```
        else if (input >= 'a' && input <= 'z')
        {
            input += key;
            if (input > 'z')
                input -= 26;
            if (input < 'a')
                input += 26;
        }
        else if (input >= '0' && input <= '9')
        output += input;
```

这部分与前两个 if 条件类似，但使用的范围是数字'0'~'9'，而不是字母'A'~'Z'。

下一行代码（它位于 if 语句的大括号内）稍有不同：

```
        else if (input >= '0' && input <= '9')
        {
            input += (keyVal % 10);
```

首先，我们使用的是用户在程序前面输入的**整数**版的密钥 keyVal；其次，使用了求模运算符（%）来确保移动量在-10 到+10 之间。

还需要检查加上 keyVal 后的结果是否位于 9 后面或 0 前面，就像检查对字母移动后是否位于 Z 后面或 A 前面一样。但不是减去 26 来回绕到字母表开头，而是需要减去 10 来确保在范围 0~9 内：

```
else if (input >= '0' && input <= '9')
            {
                input += (keyVal % 10);
                if (input > '9')
                    input -= 10;
                if (input < '0')
                    input += 10;
            }
            output += input;
```

如果将数字移动到 9 后面去了，就减去 10；如果移动到 0 前面去了，就加上 10。

现在可对要发送的消息中的字母和数字都进行加密了：

```
Enter a message to encode or decode:
Meet me at the arcade at 5pm and bring $2 to play Pac-Man :)
Enter a secret key (-25 to 25):
7
Tlla tl ha aol hyjhkl ha 2wt huk iypun $9 av wshf Whj-Thu :)
```


5pm 中的 5 移动 7 个位置并回绕到了 2，因此明文中的 5pm 在密文中变成了 2wt；而$2 中的 2 移动 7 个位置后变成了 9。要对密文进行解密，可使用密钥-7，你将得到最初的明文（包括数字）。

Secret Messages 应用的基于文本版的最终完整代码如代码清单 6-4 所示。

代码清单 6-4 最终的命令行版 Secret Messages 应用

```
import java.util.Scanner;
public class SecretMessages {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter a message to encode or decode:");
        String message = scan.nextLine();
        String output = "";
        System.out.println("Enter a secret key (-25 to 25):");
        int keyVal = Integer.parseInt(scan.nextLine());
        char key = (char) keyVal;
        for ( int x = 0; x < message.length(); x++ ) {
            char input = message.charAt(x);
            if (input >= 'A' && input <= 'Z')
            {
                input += key;
                if (input > 'Z')
                    input -= 26;
                if (input < 'A')
```

```
                    input += 26;
                }
                else if (input >= 'a' && input <= 'z')
                {
                    input += key;
                    if (input > 'z')
                        input -= 26;
                    if (input < 'a')
                        input += 26;
                }
                else if (input >= '0' && input <= '9')
                {
                    input += (keyVal % 10);
                    if (input > '9')
                        input -= 10;
                    if (input < '0')
                        input += 10;
                }
                output += input;
            }
            System.out.println(output);
            scan.close();
        }
    }
```

这个 Secret Messages 应用很有趣，可使用它在朋友之间发送加密的消息。但你的有些朋友可能没有在计算机上安装 Eclipse 或 Java JDK，要是也能够与这些朋友分享应用 Secret Messages 就好了。下一节将演示如何在不使用 Eclipse 的情况下运行命令行 Java 程序。

6.8　在不使用 Eclipse 的情况下运行命令行程序

到目前为止，我们创建了两个命令行应用，但一直都在 Eclipse 中运行它们。Eclipse 提供了便利的控制台模拟器，让我们能够看到命令行应用是什么样的，但如果要从真正的命令行（如 Windows 命令提示符或 macOS Terminal）运行应用，该如何做呢？或者我们要将命令行应用发送给没有在计算机上安装 Eclipse 的朋友，该如何做呢？

所幸大多数人都在计算机上安装了 JRE（Java 运行时环境）。

6.8.1　找到你的工作区文件夹

要运行你在 Eclipse 中编写并编译的应用，首先需要找到你的 Eclipse **工作区**文件夹。为此，请打开资源管理器（或 Finder），如图 6-7 所示。

打开工作区中的项目文件夹 SecretMessages，其中包含几个文件和文件夹，如图 6-8 所示。其中文件夹 src 包含源代码文件（扩展名为.java），而文件夹 bin 包含编译后的应用版本（扩展名为.class）。

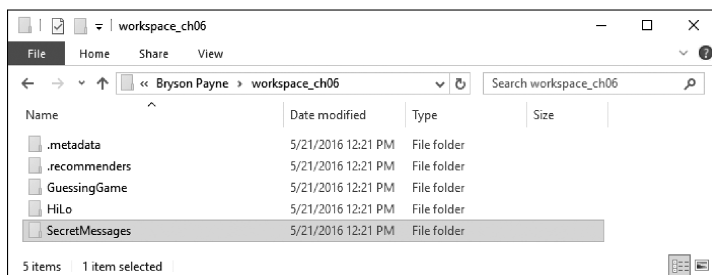


图 6-7 在我的工作区文件夹中，包含前面创建的每个项目的文件夹，还有一些 Eclipse 默认文件夹

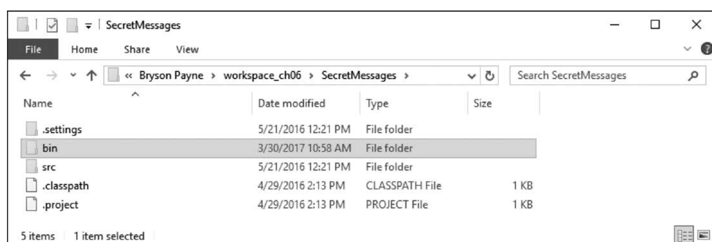


图 6-8 在项目文件夹 SecretMessages 中找到文件夹 bin

6.8.2 打开命令行窗口

接下来，打开一个命令行窗口。在 Windows 系统中，可单击“开始”按钮，再在搜索框中输入 cmd（command 的缩写）并按回车；在 macOS 系统中，可使用 Spotlight 搜索栏找到 Terminal，也可打开 Finder 并选择 Applications▶Utilities▶Terminal；在 Linux 系统中，可打开应用程序 Terminal，也可通过搜索来找到 Terminal。

在命令行或终端提示符下，需要切换到前述文件夹 bin。为此，可在终端窗口中输入 cd（表示 change directory，切换目录）和一个空格。接下来，为切换到应用的 bin 文件夹，可将这个文件夹从资源管理器或 Finder 拖放到命令提示符或终端窗口中，如图 6-9 所示。

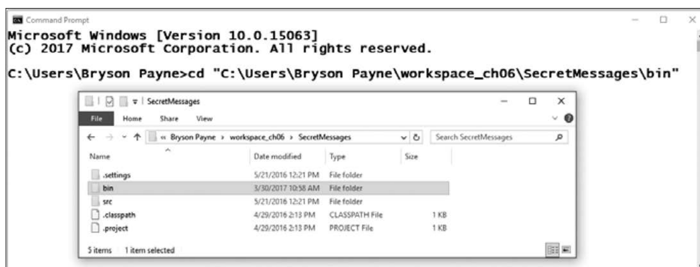


图 6-9 你可将文件夹 bin 从资源管理器或 Finder 拖放到命令行窗口，从而轻松地复制该目录路径

注意，完整的路径将出现在命令 cd 后面，它类似于下面这样（当然，你的文件夹 workspace/SecretMessages/bin 所处的位置可能不同）：

```
cd "C:\Users\Bryson Payne\workspace_ch06\SecretMessages\bin"
```

在 macOS 系统中，命令如下：

```
cd /Users/BrysonPayne/Desktop/workspace_ch06/SecretMessages/bin
```

出现这个命令后按回车键，终端将修改提示符，指出当前处于文件夹 bin 中。在 Windows 系统中，提示符类似于下面这样：

```
C:\Users\Bryson Payne\workspace_ch06\SecretMessages\bin>_
```

现在，你处于文件夹 bin 中，即编译得到的文件 SecretMessages.class 所在文件夹中。为运行字节文件 SecretMessages.class 包含的程序，请输入如下命令：

```
C:\Users\Bryson Payne\workspace_ch06\SecretMessages\bin>java SecretMessages
```

在这个命令中，拼写、大小写和空格都很重要，因此你必须准确地指定前面给 Java 类指定的名称。如果你正确地输入了命令，应用将运行，你可使用消息和密钥对其进行测试，如下所示：

```
C:\Users\Bryson Payne\workspace_ch06\SecretMessages\bin>java SecretMessages
Enter a message to encode or decode:
I'm running my app directly from the command line terminal!
Enter a secret key (-25 to 25):
12
U'y dgzzuzs yk mbb pudqofxk rday ftq oayymzp xuzq fqdyuzmx!
```

要再次运行这个应用，可重新输入命令 java SecretMessages，也可按上箭头键并按回车。图 6-10 显示了在 Windows 命令提示符中两次运行这个应用的情况，它们分别进行了加密和解密。

```
Command Prompt
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Bryson Payne>cd "C:\Users\Bryson Payne\workspace_ch06\SecretMessages\bin"

C:\Users\Bryson Payne\workspace_ch06\SecretMessages\bin>java SecretMessages
Enter a message to encode or decode:
I'm running my app directly from the command line terminal!
Enter a secret key (-25 to 25):
12
U'y dgzzuzs yk mbb pudqofxk rday ftq oayymzp xuzq fqdyuzmx!

C:\Users\Bryson Payne\workspace_ch06\SecretMessages\bin>java SecretMessages
Enter a message to encode or decode:
U'y dgzzuzs yk mbb pudqofxk rday ftq oayymzp xuzq fqdyuzmx!
Enter a secret key (-25 to 25):
-12
I'm running my app directly from the command line terminal!

C:\Users\Bryson Payne\workspace_ch06\SecretMessages\bin>
```

图 6-10　知道如何从命令行运行 Java 应用后，可在任何安装了 Java 的计算机上加密和解密消息，而不管它是否安装了 Eclipse

知道如何在计算机上运行这个应用后，可与朋友分享文件 SecretMessages.class，以便相互发

送加密后的消息。如果朋友的计算机安装了 Java（JDK 或 JRE），他们只需打开命令行程序，切换到文件 SecretMessages.class 所在的文件夹，再执行命令 `java SecretMessages`。要交换消息，只需就要使用的密钥达成一致。你可在每次加密时都使用相同的密钥，也可在与不同的朋友通信时使用不同的密钥。但别忘了，这个应用只能用来玩玩，因为任何人只要有这个程序或者有点时间，都能破解简单的凯撒加密。你将在第 7 章看到，破解凯撒加密有多容易。

6.9 小结

应用 Secret Messages 提供了一种有趣的途径，让你能够深入探索如何使用 Java 操作字符和文本字符串。在本章中，你学到了如下新技能：

- ❑ 使用凯撒加密法来加密和解密简单消息；
- ❑ 理解用于存储单个 Unicode 字符的数据类型 `char`；
- ❑ 使用方法 `charAt()`获取字符串中的特定字符；
- ❑ 使用索引或位置编号来访问字符串的特定位置；
- ❑ 使用运算符+拼接字符串及将字符与整数相加；
- ❑ 使用 `length()`确定字符串包含的字符数，再使用 `for` 循环来遍历它；
- ❑ 理解计算机如何存储字符和其他数值数据；
- ❑ 在没有安装 Eclipse 的计算机上直接从命令行运行命令行应用。

6.10 编程练习

为复习并使用学到的知识，以及获得更多的编程技能，请尝试完成这里的编程练习。如果遇到困难，可从网站 https://www.nostarch.com/learnjava/下载示例解决方案。

6.10.1 编程练习 1：Looping the Loop

我们创建了一个有趣的消息加密/解密应用，能够收发加密的文本、E-mail、推文等。在本章的第一个编程练习中，你将在应用 Secret Messages 中添加一个循环，让用户能够不断地加密和解密——想加密/解密多少次都可以。

你可始终使用相同的密钥，也可每次都让用户指定新密钥。如果每次都让用户输入密钥，用户就既能加密消息，也能解密消息。例如，输入 `8` 将要发送给朋友的消息加密，输入`-8` 将这位朋友发送给他的消息解密。

要实现这种功能，一种办法是让用户输入要加密或解密的消息，并指出要退出应用可直接按回车。然后，使用一条 `if` 语句检查用户输入了消息还是直接按了回车键。请试试吧！

提示 如果用户输入了消息，`message.length()`将大于零。

6.10.2　编程练习 2：反转并加密

我们来加大破解难度，先将消息反转，再使用凯撒加密法进行加密。这个双重加密版本结合使用了代码清单 6-1 的消息反转方法和代码清单 6-4 实现的凯撒加密功能。

将消息反转也是一种对称加密：一次加密将消息反转，再次加密将恢复到明文。结合使用消息反转和凯撒加密时，程序编写起来不会难太多，但可加大窃听者破解消息的难度。

核实你编写的程序能够正确地加密和解密消息，同时别忘了你的朋友需要这个新程序才能解密经过双重加密的消息。祝你玩得愉快！

6.10.3　编程练习 3：使用 `try` 和 `catch` 妥善地处理密钥

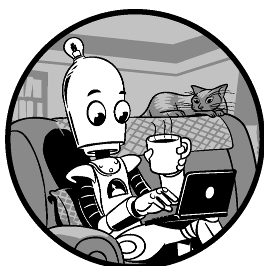
对于应用 Secret Messages，可做的最后一项改进是妥善地处理用户输入的无效密钥。还记得 `try-catch` 语句吗？

在这个应用中添加 `try-catch` 块可能有点棘手。除了添加异常处理代码，避免无效的用户无效输入（例如，在程序提示输入密钥时输入文本而不是数字）导致程序崩溃外，你还需考虑在输入无效时程序该如何做。向用户指出密钥无效并使用预定的密钥（如 `13`），还是通过使用一个循环不断地让用户输入，直到输入的密钥有效为止呢？

请尝试实现这两种方式，并选择你更喜欢的那种。

第 7 章

创建高级 GUI 并分享应用



本章将创建应用 Secret Messages 的 GUI 桌面版，我们将提供两个大型文本区域，让用户能够在 GUI 中复制并粘贴很长的消息。在本章末尾，我们将添加一个用于选择密钥的滑条（如图 7-1 所示），这让用户即便不知道密钥，也能轻松地破解凯撒加密。

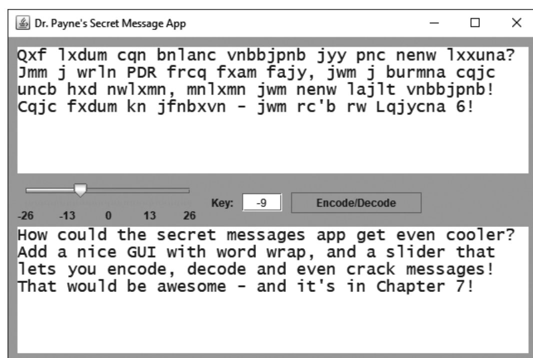


图 7-1　本章将创建的加密/解密应用 Secret Messages

相比于基于控制台的版本，这个版本对用户更友好，但我们可重用第 6 章的一些代码，因为加密和解密算法是相同的。

7.1　为创建 GUI 版 Secret Messages 应用新建一个项目

启动 Eclipse，再选择菜单 File▸New▸Java Project 来新建一个 Java 项目。将项目命名为 SecretMessagesGUI，再单击 Finish 按钮。将所有打开的文件都关闭，再在 Package Explorer 面板中展开项目文件夹 SecretMessagesGUI。右击文件夹 src，并选择 New▸Class 以新建一个 Java 源代码文件。将这个文件也命名为 SecretMessagesGUI。

这里也将使用 Swing 工具包，因此在 New Java Class 对话框中，将超类改为 `javax.swing.JFrame`，并选择指定要创建方法 `main()`的复选框。

单击 Finish 按钮，你将在文件 SecretMessagesGUI.java 中看到一些熟悉的骨架代码。在 Package Explorer 中，右击文件 SecretMessagesGUI.java，并选择 Open With▶WindowBuilder Editor 以着手为应用 Secret Messages 创建 GUI。

7.2　设计 GUI 并给组件命名

切换到 WindowBuilder Editor 的 Design 选项卡。在 Components 面板中，展开组件 `javax.swing.JFrame` 并选择 `getContentPane()`。在 Properties 面板中，将属性 `Layout` 改为 Absolute layout，如图 7-2 所示。这让我们能够以像素值指定组件的绝对位置。

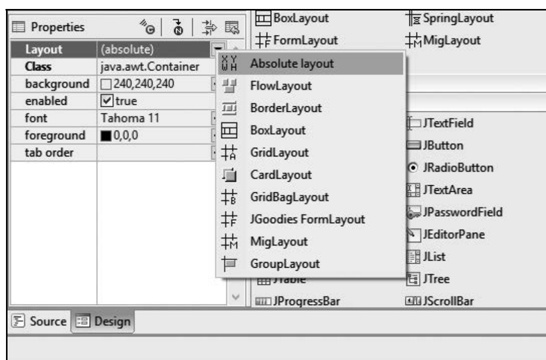


图 7-2　放置 GUI 组件前将属性 `Layout` 改为 Absolute layout

接下来，在 Components 面板中单击 javax.swing.JFrame，再在 Properties 面板中将属性 `defaultCloseOperation` 改为 EXIT_ON_CLOSE，并将属性 `title` 设置为 `*Your Name*'s Secret Message App`，如图 7-3 所示。

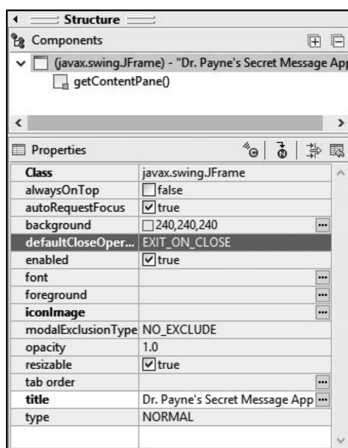


图 7-3　修改属性 `defaultCloseOperation` 和 `title` 以定制该应用

下面将 GUI 稍微增大点，让用户能够加密和解密较长的消息。确保在 Components 面板中依然选择了 javax.swing.JFrame 的情况下，在设计预览中的窗口边框外面单击，再单击窗口右下角的小型黑色大小调整框，并向右下方拖曳，将这个框架调整到 600 像素 × 400 像素，如图 7-4 所示。

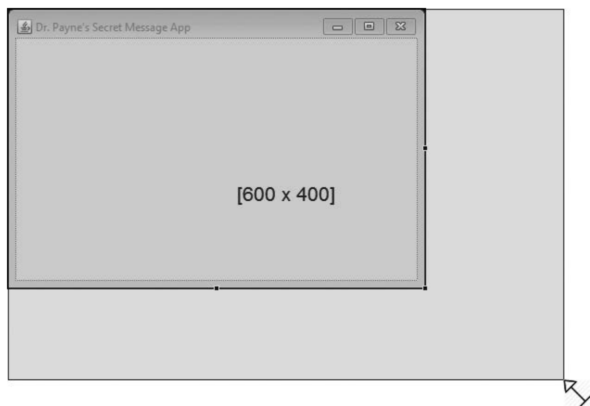


图 7-4 增大 JFrame 使其能够容纳更长的消息

现在可以开始放置 GUI 组件了。在这个 GUI 版应用中，我们要让用户能够加密和解密较长的消息，因此首先放置一个 JTextArea 组件。与 JTextField 一样，文本区域（JTextArea）让用户能够输入文本，但能够容纳和显示多行文本。

为插入供用户输入消息的 JTextArea，请在 Palette 面板的 Components 部分中单击 JTextArea，再将其放到设计预览中 JFrame 的顶部：单击并拖曳使 JTextArea 的高度大约为内容面板高度的 1/3，且边缘与框架边缘相隔很近。这个 JTextArea 应宽约 564 像素，高约 140 像素。然后，在 Properties 面板中将 Variable 属性改为 txtIn。

为创建用于输出消息的 JTextArea，右击 Preview 面板中的 txtIn 并选择 Copy，再右击内容面板并选择 Paste。将复制的 JTextArea 放在框架底部附近。为重命名这个新的 JTextArea，将 Variable 属性改为 txtOut。此时应有两个 JTextArea 组件，如图 7-5 所示。

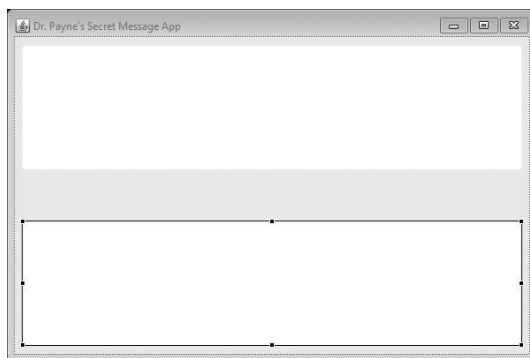


图 7-5 复制并粘贴 txtIn 以创建第二个 JTextArea，并将其命名为 txtOut

接下来，添加一个用于输入密钥值的文本框。单击 Palette 面板的 Components 部分中的 JTextField,再单击内容面板的中央附近以指定 JTextField 的位置。将这个 JTextField 的 Variable 属性改为 txtKey，并将其尺寸缩小到原来的一半左右。

在 txtKey 左边放置一个 JLabel，将其 text 属性设置为 Key:，并将其 horizontalAlignment 属性设置为 RIGHT。最后，在 txtKey 右边放置一个 JButton，将其 text 属性设置为 Encode/Decode，并加宽这个按钮让所有文本都显示出来。界面如图 7-6 所示。要在不运行应用的情况下预览 GUI，可单击 Palette 面板上方的 Test GUI 按钮，如图 7-6 中圈出的部分所示。

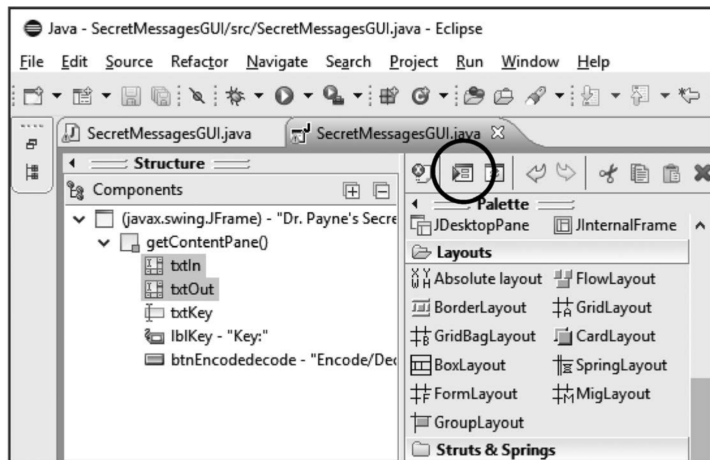


图 7-6　单击 Palette 面板上方的 Test GUI 按钮来测试 GUI

用户界面设计好后，该给程序编写代码了。

7.3　给 GUI 版 Secret Messages 应用编写代码

单击设计视图窗口左下角的选项卡 Source，切换到源代码视图，你将发现 Eclipse 在文件 SecretMessagesGUI.java 中添加了所有的 GUI 代码。为将 GUI 关联到应用，首先需要在 SecretMessagesGUI 类开头声明两个 JTextArea 变量。Eclipse 知道，JTextField 组件通常需要处理用户输入的事件处理程序，因此在类开头声明了变量 txtKey，但我们还需声明表示两个 JTextArea 组件的变量——txtIn 和 txtOut。请添加下面的最后两行代码：

```
public class SecretMessagesGUI extends JFrame {
    private JTextField txtKey;
    private JTextArea txtIn;
    private JTextArea txtOut;
```

在文件开头声明 JTextArea 变量后，需要修改构造函数 SecretMessagesGUI()中的代码——删除如下两行开头的变量类型 JTextArea：

```
    public SecretMessagesGUI() {
        setTitle("Dr. Payne's Secret Message App");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        getContentPane().setLayout(null);
        txtIn = new JTextArea();//删除行首的 JTextArea
        txtIn.setBounds(10, 11, 564, 140);
        getContentPane().add(txtIn);
        txtOut = new JTextArea();//删除行首的 JTextArea
        txtOut.setBounds(10, 210, 564, 140);
        getContentPane().add(txtOut);
```

完成这些修改后，便可以开始编写在用户单击按钮 Encode/Decode 时对消息进行加密的方法了。

7.3.1 创建方法 encode()

我们将编写的方法 encode()类似于第 3 章的猜数游戏中的 checkGuess()。方法 checkGuess()接受包含用户猜测的字符串，但不需要返回值，因此返回类型为 void。不同于 checkGuess()，encode()将接受一条消息和一个密钥，但返回加密后的消息，而不是返回 void。

方法返回值供程序使用。我们要让 encode()接受来自用户的消息和密钥，再对这些值执行代码来生成加密后的消息，供程序作为输出提供给用户。声明返回信息的方法时，我们在方法名前指定返回值的数据类型。我们要从 encode()返回一个 String 变量，因此需要将 encode()声明为 public String encode()。

我们需要告诉 Java，我们要向 encode()传递什么样的信息；这种信息被称为方法的**参数**。为给方法声明参数，我们将它们放在方法名后面的括号内：数据类型在前，参数名在后。要声明多个参数，可用逗号分隔它们。方法 encode()的声明类似于下面这样：

```
public String encode( String message, int keyVal )
```

下面来添加用于将方法体括起的大括号，并将方法 encode()放在两个 JTextArea 变量的声明和构造函数 SecretMessagesGUI()之间：

```
public class SecretMessagesGUI extends JFrame {
    private JTextField txtKey;
    private JTextArea txtIn;
    private JTextArea txtOut;
    public String encode( String message, int keyVal ) {
    }
    public SecretMessagesGUI() {
```

Eclipse 将给方法 encode()加上红色下划线，让你知道它没有返回值，但这是因为我们还未在方法体内编写代码。在这里，我们将重用第 6 章的一些代码。

声明 String 变量 output，并将其初始化为空字符串。再添加一行代码，它使用 return 语句将 output 作为方法 encode()的结果返回：

```
public String encode( String message, int keyVal ) {
    String output = "";
    return output;
}
```

由于 encode()被声明为返回一个 String 值，而我们返回了 String 变量 output，因此 Eclipse 将删除红色下划线，让我们知道缺失返回值的问题已解决。

接下来，我们将重用基于文本的应用版本中加密消息的代码。为此，在 Project Explorer 面板中打开第 6 章创建的项目 SecretMessages。

我们可复制从 char key 到 for 循环的右大括号(程序末尾附近的 System.out.println(output); 语句前面) 的代码，它们使用凯撒加密算法对消息进行加密。

将从 SecretMessages.java 复制的代码粘贴到 SecretMessagesGUI.java 中，放在方法 encode() 中我们刚编写的两行代码之间。最终的方法 encode()类似于下面这样：

```
public String encode( String message, int keyVal ) {
    String output = "";
    char key = (char) keyVal;
    for ( int x = 0; x < message.length(); x++ ) {
        char input = message.charAt(x);
        if (input >= 'A' && input <= 'Z')
        {
            input += key;
            if (input > 'Z')
                input -= 26;
            if (input < 'A')
                input += 26;
        }
        else if (input >= 'a' && input <= 'z')
        {
            input += key;
            if (input > 'z')
                input -= 26;
            if (input < 'a')
                input += 26;
        }
        else if (input >= '0' && input <= '9')
        {
            input += (keyVal % 10);
            if (input > '9')
                input -= 10;
            if (input < '0')
                input += 10;
        }
        output += input;
    }
    return output;
}
```

由于使用的变量名相同，因此可重用实现凯撒加密的代码。以一致的方式命名是个不错的习惯，这样可在多个平台中重用编写良好的 Java 代码。

7.3.2 给按钮 Encode/Decode 编写事件处理程序

我们要让用户在 GUI 中提供输入消息和密钥，以便将它们传递给 encode()，还要在用户单击 Encode/Decode 按钮时返回输出消息，因此下面来编写处理按钮单击事件的代码。

切换到 SecretMessagesGUI.java 的设计视图，并双击按钮 Encode/Decode。Eclipse 将切换到源代码视图，并给按钮 Encode/Decode 添加事件处理程序 actionPerformed()，如下所示：

```
JButton btnEncodedecode = new JButton("Encode/Decode");
btnEncodedecode.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
    }
});
```

这是一个匿名内部类，与我们在第 3 章遇到的相似。我们只需在 actionPerformed()中添加代码，告诉 Java 在用户单击这个按钮时该如何做。

我们要让按钮 Encode/Decode 执行如下步骤。

(1) 从 txtIn 中获取输入消息。

(2) 从 txtKey 中获取密钥。

(3) 使用密钥对消息进行加密。

(4) 在 txtOut 中显示输出消息。

请想想如何完成这些步骤，并尝试提供解决方案，再接着往下读，看看我提供的解决方案有何不同。

对于第 1 步，可创建一个名为 message 的 String 变量，用于存储 txtIn 中的文本。与 JTextField 一样，JTextArea 也有方法 getText()：

```
public void actionPerformed(ActionEvent arg0) {
    String message = txtIn.getText();
```

对于第 2 步，可使用方法 Integer.parseInt()来获取用户在 txtKey 中输入的密钥；这个方法从字符串中提取一个整数值。然后，将获得的整数值存储到变量 key 中：

```
String message = txtIn.getText();
int key = Integer.parseInt( txtKey.getText() );
```

要完成第 3 步（加密消息），只需调用方法 encode()并向它传递两个实参。**实参**是提供给方法的形参的值。别忘了，前面将 encode()定义为接受两个形参，因此我们需要向 encode()传递两个实参——变量 message 和 key。

可使用语句 encode(message, key)来调用 encode()。执行完毕后，这个方法将返回加密后的消息——变量 output。请注意，方法返回的是值本身，而不是存储它的变量。这意味着虽然加密后的消息存储在变量 output 中，但当 encode()返回 output 时，我们实际上获得的是加密后的字符串。另外，在 encode()内创建的代码和变量都不能进入程序的其他部分，因此方法 encode()

返回后，变量 output 就不再存在。如果我们要保存这个方法返回的值，就必须将其存储到一个变量中。为保持命名的一致性，我们将这个变量也命名为 output：

```
String message = txtIn.getText();
int key = Integer.parseInt( txtKey.getText() );
String output = encode( message, key );
```

最后，使用方法 setText()在 txtOut 中显示输出消息：

```
String message = txtIn.getText();
int key = Integer.parseInt( txtKey.getText() );
String output = encode( message, key );
txtOut.setText( output );
```

方法 actionPerformed()差不多编写好了，但还需添加一些错误处理逻辑。我们已经使用了方法 Integer.parseInt()，它在用户输入无效时会引发异常，因此我们需要添加 try-catch 语句。

7.3.3　处理无效输入和用户错误

在处理按钮单击的代码中，调用了 Integer.parseInt()，而第 3 章说过，这个方法会在用户输入无效时引发异常。具体地说，如果用户在 txtKey 中没有输入任何内容或输入的不是整数，方法 Integer.parseInt()将失败。

我们需要使用 try-catch 块来处理异常，从而避免无效输入导致程序崩溃。我们只想在用户正确地输入了密钥时对消息进行加密或解密，因此可将 actionPerformed()内的全部 4 行代码都放在 try 块内：在第一行前添加关键字 try 和左大括号，并在第 4 行后面添加右大括号，如下所示：

```
public void actionPerformed(ActionEvent arg0) {
    try {
        String message = txtIn.getText();
        int key = Integer.parseInt( txtKey.getText() );
        String output = encode( message, key );
        txtOut.setText( output );
    } catch (Exception ex) {
    }
}
```

Eclipse 将以红色显示右大括号，因此我们需要接着添加 catch 块，如上所示。我们需要告诉 Java 在输入无效时如何做，我们将在 catch 语句的大括号内指出这一点。但在此之前，先来编写方法 main()，以便能够测试应用。

7.3.4　编写方法 main()并运行应用

与第 3 章创建猜数游戏一样，需要在方法 main()中添加一些启动代码，以创建 SecretMessagesGUI 实例、正确地设置 GUI 窗口的尺寸以及让 GUI 可见。为此，在文件 SecretMessagesGUI.java

末尾附近找到 Eclipse 提供的方法 main()的存根，并添加如下 3 行代码：

```
public static void main(String[] args) {
❶ SecretMessagesGUI theApp = new SecretMessagesGUI();
❷ theApp.setSize(new java.awt.Dimension(600,400));
❸ theApp.setVisible(true);
}
```

❶处的代码行创建一个名为 theApp 的 SecretMessagesGUI 对象。后面跟着类名的关键字 new 调用构造函数 SecretMessagesGUI()，后者创建 GUI 的所有组件。

接下来，设置 JFrame 的尺寸，使其与我们设计布局时指定的宽度和高度一致，即 600 像素 × 400 像素❷。最后，将 JFrame 的 visible 属性设置为 true❸，让用户能够看到 GUI。请保存所做的修改，再运行这个应用以测试它，如图 7-7 所示。

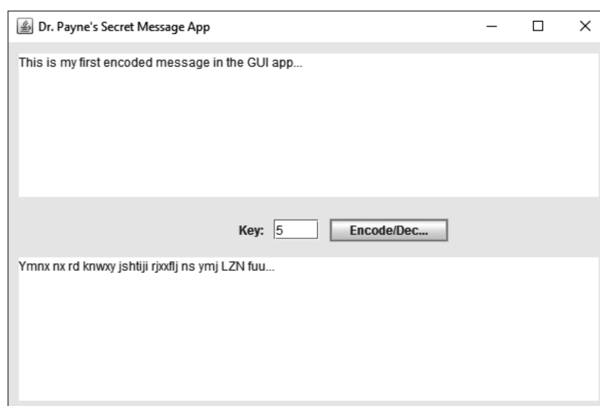


图 7-7 应用 Secret Messages 朴实无华，但能够加密和解密消息

到目前为止，文件 SecretMessagesGUI.java 的完整源代码如下所示：

```
import javax.swing.JFrame;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.JLabel;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
public class SecretMessagesGUI extends JFrame {
    private JTextField txtKey;
    private JTextArea txtIn;
    private JTextArea txtOut;
    public String encode( String message, int keyVal ) {
        String output = "";
        char key = (char) keyVal;
        for ( int x = 0; x < message.length(); x++ ) {
            char input = message.charAt(x);
```

```
            if (input >= 'A' && input <= 'Z')
            {
                input += key;
                if (input > 'Z')
                    input -= 26;
                if (input < 'A')
                    input += 26;
            }
            else if (input >= 'a' && input <= 'z')
            {
                input += key;
                if (input > 'z')
                    input -= 26;
                if (input < 'a')
                    input += 26;
            }
            else if (input >= '0' && input <= '9')
            {
                input += (keyVal % 10);
                if (input > '9')
                    input -= 10;
                if (input < '0')
                    input += 10;
            }
            output += input;
        }
        return output;
    }
    public SecretMessagesGUI() {
        setTitle("Dr. Payne's Secret Message App");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        getContentPane().setLayout(null);
        txtIn = new JTextArea();
        txtIn.setBounds(10, 11, 564, 140);
        getContentPane().add(txtIn);
        txtOut = new JTextArea();
        txtOut.setBounds(10, 210, 564, 140);
        getContentPane().add(txtOut);
        txtKey = new JTextField();
        txtKey.setBounds(258, 173, 44, 20);
        getContentPane().add(txtKey);
        JLabel lblKey = new JLabel("Key:");
        lblKey.setBounds(202, 176, 46, 14);
        getContentPane().add(lblKey);
        JButton btnEncodedecode = new JButton("Encode/Decode");
        btnEncodedecode.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                try {
                    String message = txtIn.getText();
                    int key = Integer.parseInt( txtKey.getText() );
                    String output = encode( message, key );
                    txtOut.setText( output );
                } catch (Exception ex) {
                }
```

```
            }
        });
        btnEncodedecode.setBounds(312, 172, 144, 23);
        getContentPane().add(btnEncodedecode);
    }
    public static void main(String[] args) {
        SecretMessagesGUI theApp = new SecretMessagesGUI();
        theApp.setSize(new java.awt.Dimension(600,400));
        theApp.setVisible(true);
    }
}
```

这是应用 Secret Messages 的第一个版本，它虽然管用，但在外观方面还有改进的空间。只需做几个细微的调整，就可让 GUI 看起来更专业。

7.4　改进 GUI

我们切换到设计视图，以消除前一节指出的问题。首先，单击按钮 Encode/Decode，并将其加宽些。其他计算机默认使用的字体和字号可能不同，所以为安全起见，需要让这个按钮提供一点点额外的空间。

下面来增大两个 JTextArea 的字号，让用户更容易看清楚。我们要修改这两个文本区域的 font 属性，因此可同时选择 txtIn 和 txtOut：单击 txtIn，再按住 CTRL（或⌘）键并单击 txtOut。

选择这两个文本区域后，在 Properties 面板中点击 font 属性值右边的三个点，这将打开 Font Chooser 对话框，如图 7-8 所示。



图 7-8　同时修改两个 JTextArea 的 font 属性

将字号设置为 18 或更大的值，并选择一种你喜欢的字体。请始终使用你熟悉的字体，这样与朋友分享应用时，应用的外观就不会发生太大的变化。我选择的是 Lucida Console 字体，并将字号设置为 18 点。设置好字体属性后，单击 OK 按钮让设置生效。

最后，修改内容面板的背景色，使其更具个性。请选择内容面板：在左上角的 Components 面板中单击 `getContentPane()`。然后，在 Properties 面板中，单击属性 `background` 旁边的三个点。在打开的 Color Chooser 对话框中，单击选项卡 Named colors，如图 7-9 所示。你将看到很多颜色，请选择你喜欢的颜色，再单击 OK 按钮。

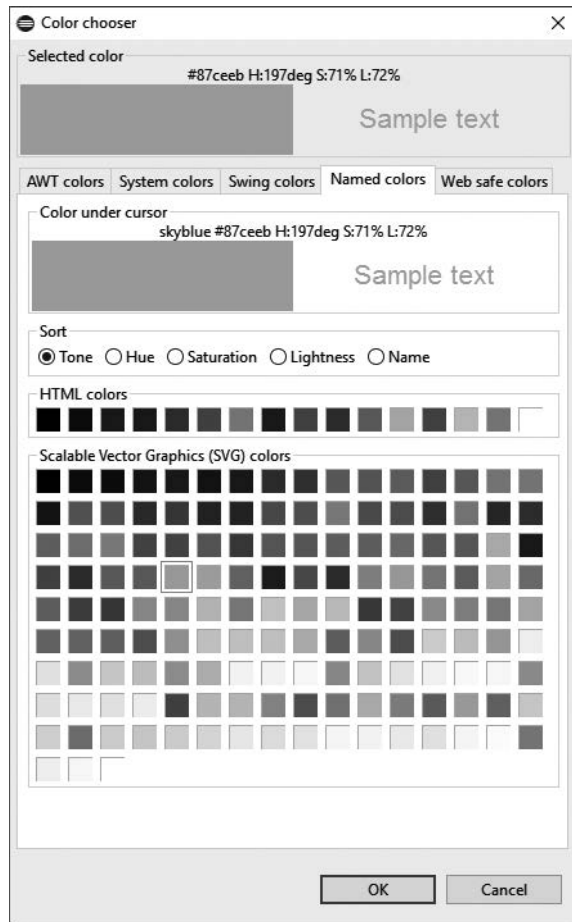


图 7-9　修改内容面板和按钮的背景色，让应用显得更时尚

你还可修改按钮 Encode/Decode 的背景色。在设计预览中可能无法看到所有的修改，但只要运行应用，就能看到所有的设置，包括新的背景色，如图 7-10 所示。



图 7-10 修改按钮尺寸和字号后的 GUI

你还可根据喜好定制众多其他的属性。请依次单击每个组件，并在 Properties 面板中尝试调整其属性，直到对应用的外观满意为止。别忘了保存应用，以免所做的修改消失殆尽。

7.4.1 设置换行和折词

本书前面的示例中使用的消息都不超过一行，但如果输入多行文本，结果将如何呢？

我们来尝试在输入文本区域中输入或粘贴一个很长的句子。这里使用美国总统亚伯拉罕·林肯葛底斯堡演说的第一句话：

> Four score and seven years ago our fathers brought forth on this continent, a new nation, conceived in Liberty, and dedicated to the proposition that all men are created equal.

你可使用这个句子，也可自己编写很长的句子。无论如何选择，当你输入句子后，都将发现一个问题，如图 7-11 所示。



图 7-11 消息较长时不会自动换行

要正确地在单词之间换行，需要在 Properties 面板中修改每个文本区域的两个属性。请再次在设计预览中同时选择两个 `JTextArea` 组件，再在 Properties 面板中选择 `lineWrap` 和 `wrapStyleWord` 旁边的复选框。其中第一个属性（`lineWrap`）让 Java 将超过一行的文本换行，而第二个属性（`wrapStyleWord`）让 Java 在单词之间换行，就像你在字处理程序中所做的或在

本书中所看到的一样。

现在保存并运行应用。将长句子粘贴到第一个文本区域中，你将发现情况有了改善。现在，文本自动排成了多行，且每行包含的单词都是完整的。请输入密钥并单击 Encode/Decode 按钮，你将看到输出文本区域内的文本也以同样的方式换行，如图 7-12 所示。

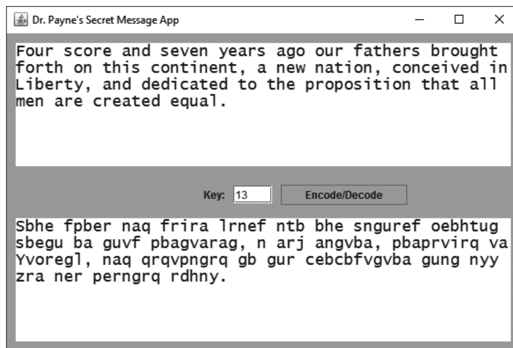


图 7-12　将属性 lineWrap 和 wrapStyleWord 都设置为 true 后，文本分成了多行，且分行位置在单词之间

这个版本看起来更专业了，但如果用户在文本框中输入无效的密钥，结果将如何呢？什么都不会发生。前面添加了一条 try 语句，但没有完整地编写 catch 语句。下面就来完成这项工作：使用弹出框将错误告知用户。

7.4.2　处理无效输入和用户错误：第 2 部分

本章前面将按钮事件处理程序放在了一个 try 块中，但 catch 语句体还是空的，现在该在其中添加代码了。

catch 语句将捕获文本框 txtKey 中没有输入或输入无效的情况。如果用户忘记输入密钥或输入的值不是数字，一种处理办法是显示一个指出问题的消息框，如图 7-13 所示。

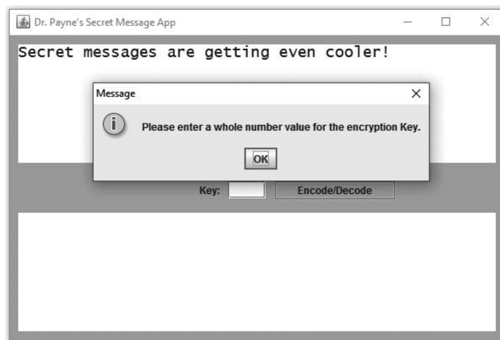


图 7-13　将错误告知用户的弹出框

另一个有用的润色是，在用户关闭弹出框后，将光标移到文本框 txtKey 并选择其中所有的文本，让用户可轻松地输入新值，而无需手动单击并选择原来的值。

前面使用的 JFrame 和 JTextField 等 GUI 组件都属于工具包 javax.swing，这个工具包含提供了处理各种弹出窗口的类。这些弹出窗口包括**输入对话框**、**确认对话框**和**消息框**，它们让用户能够提供文本输入、选择 Yes/No/OK 或 Cancel 以及向用户显示信息。

为向用户显示错误消息，最佳的方式是使用消息框。javax.swing.JOptionPane 类有一个名为 showMessageDialog()的方法，它接受两个参数：父窗口和要显示的消息。编写大型应用程序时，可使用父窗口参数让对话框显示在主窗口中央。但就这个应用而言，我们可将这个参数设置为关键字 null，让对话框显示在桌面中央，如下所示：

```
JOptionPane.showMessageDialog(null,
      "Please enter a whole number value for the encryption key.");
```

这行代码弹出一个内容为“Please enter a whole number value for the encryption key. ”的消息框。

为将这行代码添加到前面创建的 catch 语句中，请在设计视图中双击按钮 Encode/Decode。这是一种快捷方式，让 Eclipse 将你直接带到按钮单击事件处理程序的源代码处，其中包含前述 catch 语句。

显示错误消息后，我们还要将光标放到文本框 txtKey 中并选择其中所有的文本，就像 GUI 版猜数游戏中一样。我们将在 catch 语句的大括号中，添加显示消息框的代码以及猜数游戏中使用的 requestFocus()和 selectAll()语句。Encode/Decode 按钮的事件处理程序的最终代码类似于下面这样：

```
        btnEncodedecode.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                try {
                    String message = txtIn.getText();
                    int key = Integer.parseInt( txtKey.getText() );
                    String output = encode( message, key );
                    txtOut.setText( output );
                } catch (Exception ex) {
                    JOptionPane.showMessageDialog(null,
    "Please enter a whole number value for the encryption key.");
                    txtKey.requestFocus();
                    txtKey.selectAll();
                }
            }
        });
```

注意 添加 JOptionPane 时，如果没有使用内容助手，或者在 JOptionPane 处出现错误了，可按 CTRL-Shift-O 在代码开头导入 javax.swing.JOptionPane 类。

现在运行应用，并保留文本框 txtKey 为空或在其中输入非数字文本。这将出现消息框，让你输入有效的密钥。另外，还将选中文本框 txtKey 的内容，让你关闭消息框后可轻松地输入整数。

用户刚开始可能不知道该在文本框 txtKey 中输入什么，因此指定默认值可能有所帮助。为此，请单击 Design 选项卡，选择文本框 txtKey，并在 Properties 面板中将属性 text 指定为默认的密钥。我将默认密钥设置为 3，但你可选择任何数字。另外，将文本框 txtKey 的 horizontalAlignment 属性改为 CENTER。你可随便修改其他属性，如字体颜色和样式，让文本框 txtKey 变成你喜欢的样子。

7.4.3　添加滑条

下面再对应用 Secret Messages 的界面做一项改进：添加一个数字滑条，让用户能够快速调整密钥值，并查看使用不同密钥值的结果。

切换到设计视图，并在 Palette 面板的 Components 部分选择 JSlider 组件。将鼠标指向设计预览左边的中央（提示用户输入密钥的标签旁边），再单击将 JSlider 放在如图 7-14 所示的地方——如果必要，可在放置组件后再调整其位置。

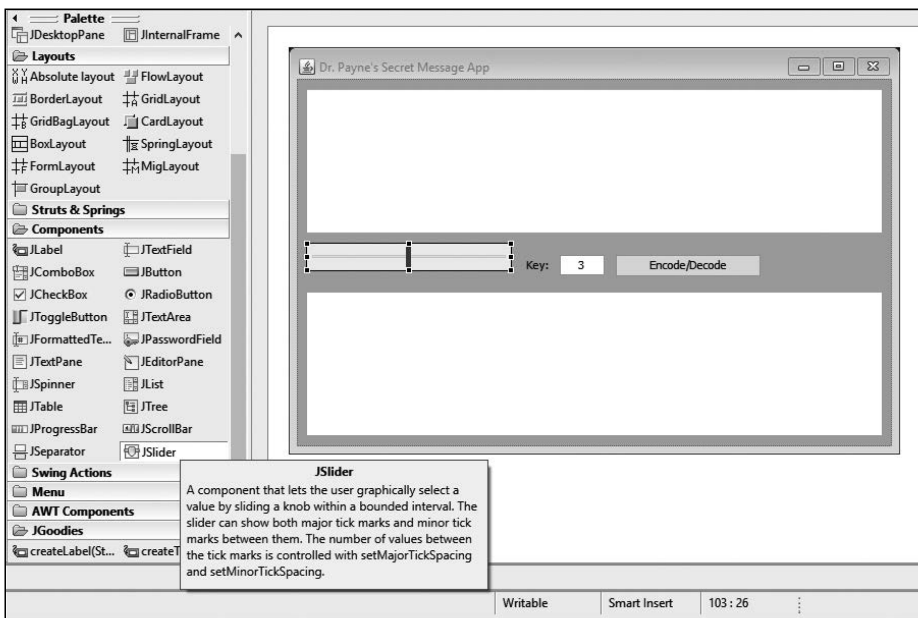


图 7-14　在图形用户界面中添加一个 JSlider，让用户能够轻松地尝试使用不同的密钥值

下面来定制 JSlider 的一些属性。首先，修改其背景色，使其与应用的其他部分匹配。在设计预览中选择了 JSlider 的情况下，在 Properties 面板中单击属性 background 旁边的三个点，然后在打开的 Color Chooser 对话框中单击选项卡 Named colors，并选择给 GUI 的其他部分指定的背景色。单击 OK 按钮保存滑条的背景色。

接下来，给滑条添加自定义标签、刻度和默认值。首先，启用属性 `paintLabels`——选择它旁边的复选框 true。其次，将属性 `maximum` 和 `minimum` 分别设置为 26 和-26，让用户能够在正确的范围内选择密钥值。

接下来，将属性 `minorTickSpacing` 设置为 1，这将在滑条中添加小刻度，让用户明白可能的取值范围。然后，将属性 `majorTickSpacing` 设置为 13，这将在滑条上显示范围-26 到 26 内所有 13 的倍数。现在启用属性 `paintTicks`——选择它旁边的复选框 true。完成这些修改后，Properties 面板如图 7-15 所示。

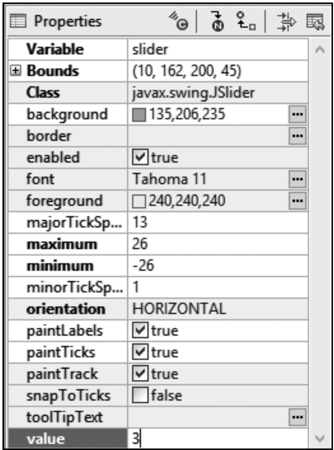


图 7-15 定制滑条的属性

就要大功告成了！请修改滑条的默认值，使其与文本框 `txtKey` 中的默认密钥值相同。为此，可修改属性 `value`。我选择的默认密钥值为 3，因此我将滑条的属性 `value` 设置为 3。最后，需要稍微加大滑条的高度，让标签显示出来。最终的滑条如图 7-16 所示。

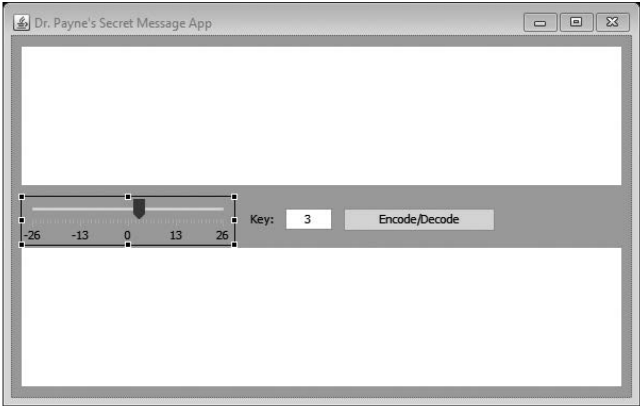


图 7-16 滑条就绪后应用 Secret Messages 的 GUI

现在，需要添加让滑条发挥作用的代码，让用户能够轻松地选择不同的密钥值，从而快速地加密和解密消息。

7.5　添加让滑条起作用的代码

我们要让用户能够单击并拖曳前一节添加的滑条来修改密钥值，因此接下来将添加一个监听滑条值变化的事件处理程序。

请右击（或按住 Control 键并单击）设计预览中的滑条，并选择 Add event handler▸change▸stateChanged，如图 7-17 所示。事件处理程序 `stateChanged` 的工作原理类似于按钮 Encode/Decode 的事件处理程序 `actionPerformed`，但在用户调整滑条的位置以修改密钥值时运行。

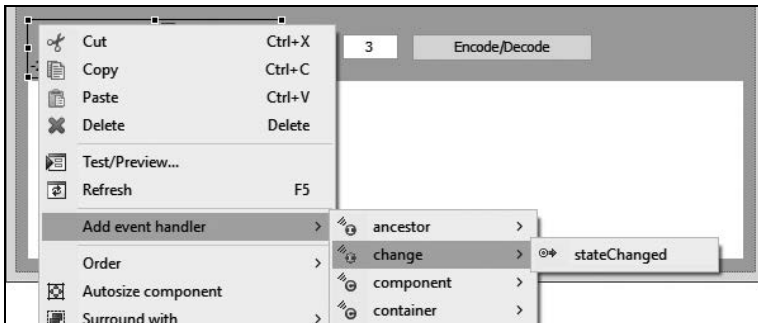


图 7-17　给滑条添加一个事件处理程序，以检测用户修改滑块值的操作

当你通过单击给滑条添加 `stateChanged` 事件处理程序时，Eclipse 将自动生成一个匿名内部类，这个匿名内部类与 Eclipse 为按钮 Encode/Decode 创建的匿名内部类很像。查看这个匿名内部类前，我们先在类开头（`JTextArea` 变量声明后面）声明一个 `JSlider` 变量。为此，需要将 `JSlider` 声明为一个实例变量。请在 `SecretMessagesGUI` 类开头的变量声明处添加下面显示的最后一行代码：

```
public class SecretMessagesGUI extends JFrame {
    private JTextField txtKey;
    private JTextArea txtIn;
    private JTextArea txtOut;
    private JSlider slider;
```

然后，向下滚动到滑块的代码处。别忘了，可随时切换到事件处理程序处。为此可切换到设计视图，再通过右击再次添加事件处理程序。现在，修改第一行——将类型声明 `JSlider` 删除，如下所示：

```
        slider = new JSlider();
        slider.addChangeListener(new ChangeListener() {
            public void stateChanged(ChangeEvent arg0) {
```

我们要让滑条采取的第一项措施是，更新文本框 `txtKey` 使其显示滑条的当前位置。要获取

滑条的当前位置，可使用方法 getValue()；而要设置文本框 txtKey 中的文本，可使用方法 setText()。知道这些后，就可编写方法 stateChanged()的第一行代码了：

```
public void stateChanged(ChangeEvent arg0) {
    txtKey.setText( "" + slider.getValue() );
```

方法 slider.getValue()返回一个整数——滑条的当前位置值，因此我们将一个空字符串与这个整数相加，从而将其转换为字符串（文本值）。这行代码让文本框 txtKey 显示滑条的值，但不会自动使用新密钥重新加密消息。如果你此时运行这个应用，依然需要单击 Encode/Decode 按钮来加密消息。

下面来修改方法 stateChanged()的代码，以便在用户调整滑条时重新加密消息，就像用户单击了按钮 Encode/Decode 一样。为此，可复制按钮 Encode/Decode 的 try 语句内的代码，并将其粘贴到方法 stateChanged()的第一条语句后面，再修改一个地方：

```
slider = new JSlider();
slider.addChangeListener(new ChangeListener() {
    public void stateChanged(ChangeEvent arg0) {
        txtKey.setText( "" + slider.getValue() );
        ❶ String message = txtIn.getText();
        ❷ int key = slider.getValue();
        ❸ String output = encode( message, key );
        ❹ txtOut.setText( output );
    }
});
```

❶❸和❹处的代码行是直接从按钮 Encode/Decode 的事件处理程序中复制而来的，但我们在❷处做了细微的修改：不从文本框中获取值，而是使用方法 getValue()直接访问滑条的值（一个整数）。这意味着不需要 try-catch 语句，因为我们不再依赖用户手动输入的值。这对用户来说更方便，而对程序员来说更安全，因为使用滑条避免了可能发生的错误，同时让界面使用起来更容易。

请保存所做的修改并运行应用，你将能够输入消息，并调整滑条的位置来尝试不同的密钥值。为测试解密功能，使用 CTRL-C（或⌘-C）复制下面那个文本区域中加密后的消息，再使用 CTRL-V（或⌘-V）将其粘贴到上面的文本区域中，然后左右移动滑条，直到在下面的文本区域中看到原来的消息。

即便不知道密钥，你也能够破解凯撒加密。为此，你可缓慢地移动滑条，直到能够读懂下面的文本区域显示的解密后的消息。下面就来试一试！在应用上面的文本区域中输入下面的消息：

```
Epdetyr esp lmtwtej ez mcplv esp Nlpdlc ntaspc htes xj Dpncpe Xpddlrp laa...
```

然后，将滑条移到最左边，再缓慢地向右移动，直到出现的消息是可以读懂的。你知道加密这条消息时使用的密钥是什么吗？请在图 7-18 中查找线索。

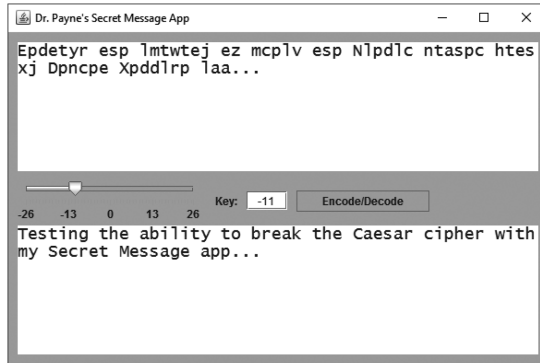


图 7-18　可使用应用 Secret Messages 来破解凯撒加密，方法是左右移动滑条直到出现明文！

将滑条移到-11 处时出现了明文，因此密钥必定是 11。你可能还注意到了，使用密钥 15 也可破解这条消息，这是因为 15 = 26 – 11。对于使用基本拉丁字母表的消息，总是有两个密钥可用来破解。当然，你也可发送使用其他语言编写的消息，如图 7-19 所示。

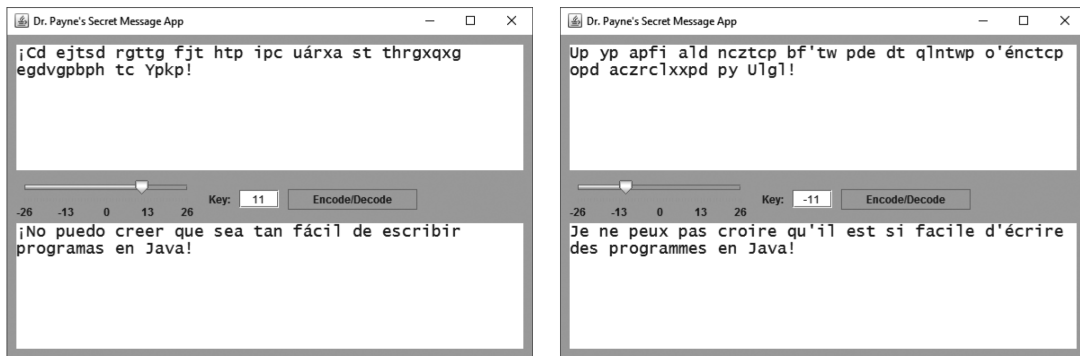


图 7-19　这个版本的 Secret Messages 应用可用来加密和解密使用任何语言编写的消息，只要它只使用基本拉丁字母表中的字符；这里显示的是西班牙语（左）和法语（右）

你可复制并粘贴加密后的消息，并通过 E-mail、Twitter 和 Facebook 将其发送给朋友；你甚至可发送加密后的短信，但这在移动设备中更容易。第 8 章将介绍如何将这个桌面版 Secret Messages 应用转换为移动版。

要是能够轻松地与朋友分享 Secret Messages 应用就好了，这样就能相互发送加密的消息了。下一节将介绍如何分享。

7.6　以可运行的 JAR 文件的方式分享应用

要让应用 Secret Messages 发挥作用，需要能够与朋友分享它——即便他们不知道如何使用 Java 编写代码，也没有在计算机上安装 Eclipse。

好消息是 Eclipse 和 Java 让你能够轻松地将应用导出为可执行的文件，并与任何运行 Java 的计算机分享。你的朋友无需下载 Java JDK 开发人员版，而只需有 JRE——大多数计算机都安装了。Eclipse 可将应用导出为可运行的 Java 归档（JAR）文件，而你可通过 E-mail、U 盘或其他方式分享这个文件。你的朋友只需双击这个 JAR 文件就能运行应用。

在 Eclipse 中，要导出可执行的 JAR，可选择菜单 File▶Export，再展开文件夹 Java 并单击 Runnable JAR file，如图 7-20 所示。

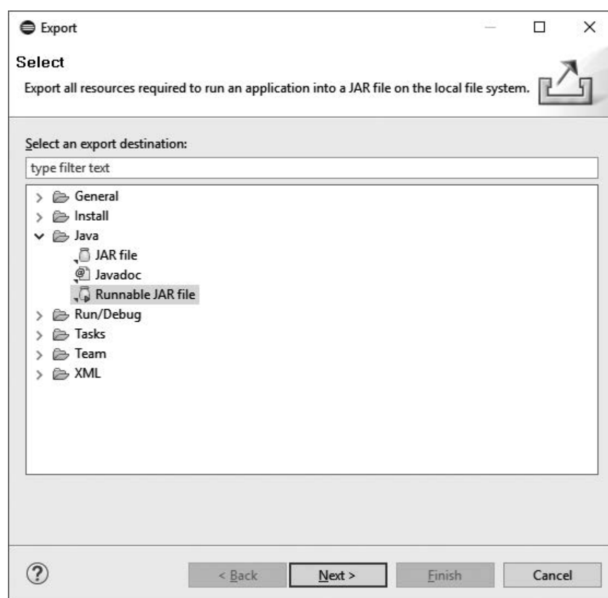


图 7-20　通过导出可执行的 JAR 文件来分享应用

单击 Next 按钮，Eclipse 将让你指定启动配置和导出目的地。所谓启动配置，指的是要运行哪个项目中的哪个类或应用。在 Launch configuration 部分，单击下拉列表并选择 SecretMessagesGUI – SecretMessagesGUI，这意味着你要运行项目 SecretMessagesGUI 中的文件 SecretMessagesGUI.class。

注意　仅当应用至少被编译并运行了一次后，它才有启动配置。

导出目的地指的是可执行的应用的文件名和存储位置。在 Export destination 部分，单击 Browse 按钮，指定要将最终的应用存储到哪个文件夹，如 Desktop；再给程序文件指定名称，如 Secret Messages.jar。给文件命名时，可使用空格或其他特殊字符，让文件名为你希望别人看到的样子。指定文件夹和文件名后单击 Save 按钮，再单击 Finish 按钮，如图 7-21 所示。

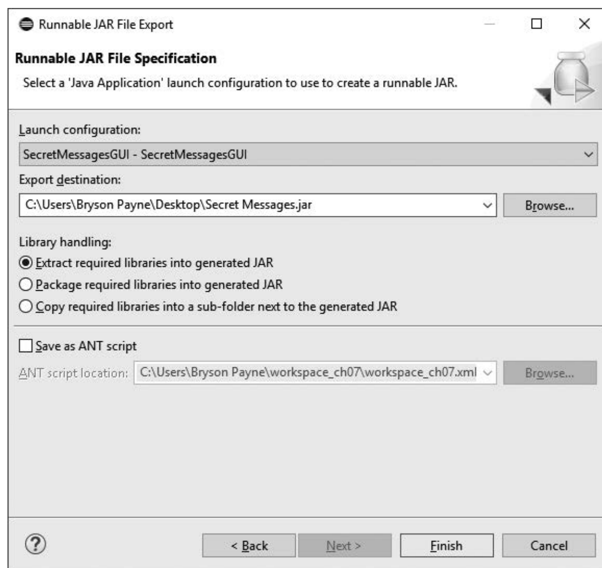


图 7-21　给可执行的 JAR 文件指定存储位置和名称

单击 Finish 按钮后可能出现警告，你可不管它，直接单击 OK 按钮。进入 JAR 文件的存储位置，你将看到 Java 图标和前面指定的文件名，如图 7-22 所示。现在，你应该能够运行这个程序并收发加密后的消息。

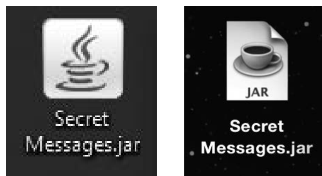


图 7-22　Windows（左）和 macOS（右）系统的桌面上可执行 JAR 文件的图标

注意　在较新版的 macOS 系统中，首次运行该应用时需要这样做：按住 Control 键并单击该应用，再选择 Open，然后在出现的对话框中单击 Open。

出于安全考虑，有些 E-mail 程序可能禁止将扩展名为.jar 的文件作为附件，但你可将文件上传到文件分享网站（如 Dropbox、Google Drive 或 SkyDrive），再在这里分享文件。获得这个文件后，朋友只需运行它，就可与你分享加密后的消息。

注意　你也可将第 3 章创建的 GUI 版猜数游戏导出为可执行的 JAR 文件。为此，只需对项目 GuessingGame 中的文件 GuessingGame.java 执行前面介绍的步骤。

7.7 小结

你在本章的应用中，重用了基于文本的应用 Secret Messages 中的代码，并加深了对 GUI 设计和编程的理解。下面是本章介绍的一些技能：

- 以一致的方式给 GUI 组件和变量命名，以提高代码的可读性和可重用性；
- 重用算法，如从基于文本的应用版本中复制的方法 `encode()`；
- 声明接受参数并返回信息的方法；
- 为各种 GUI 组件（如按钮、滑条和文本框）编写复杂的事件处理程序；
- 在设计视图中修改 GUI 组件的属性，以进一步定制 UI；
- 为 `JTextArea` 组件设置换行和折词；
- 弹出包括消息框在内的各种 `javax.swing.JOptionPane` 对话框；
- 在 GUI 中添加并设置滑条（`JSlider`）；
- 使用滑条的 `stateChanged()`事件处理程序修改文本框中的文本；
- 导出可执行的 JAR 文件以便与朋友分享应用。

7.8 编程练习

为复习并使用学到的知识，以及获得更多的编程技能，请尝试完成这里的编程练习。如果遇到困难，可从本书的配套网站（https://www.nostarch.com/learnjava/）下载示例解决方案。

7.8.1 编程练习 1：自动移动加密后的消息

应用 Secret Messages 非常适合用来将简单的加密消息发送给朋友，但你也可自己使用。在这个应用中，用户经常需要执行的一项任务是，复制输出文本区域中的加密消息并将其粘贴到输入文本区域中。在这个编程练习中，你将创建一个 Move Up ^按钮，它将加密消息移到输入文本区域中并自动进行解密！

请将按钮 Move Up ^放在按钮 Encode/Decode 旁边，再添加一个事件处理程序，它获取文本区域 `txtOut` 中的文本，并将其放到文本区域 `txtIn` 中。另外，让按钮 Move Up ^的事件处理程序将滑条的值取负。你还可以修改这个按钮的背景色，使其与应用的其他部分匹配。图 7-23 提供了一个示例解决方案。

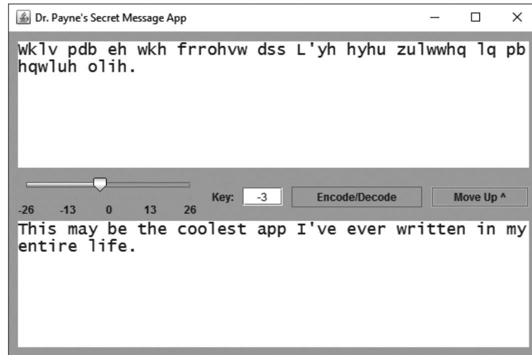


图 7-23 添加自动解密消息的按钮 Move Up ^，它将加密消息移到输入文本区域并对滑条的值取负

7.8.2　编程练习 2：添加滚动功能

对这个应用可做的另一项改进是，让其能够处理更长的消息。为此，可给输入文本区域加上滚动条，并在用户输入的消息太长时自动向下滚动。JTextArea 本身不会添加滚动条，但 Eclipse 让你能够轻松而快捷地给任何文本区域添加滚动条。

在设计视图中，右击其中一个 JTextArea 并选择 Surround with▶javax.swing.JScrollPane，如图 7-24 所示。

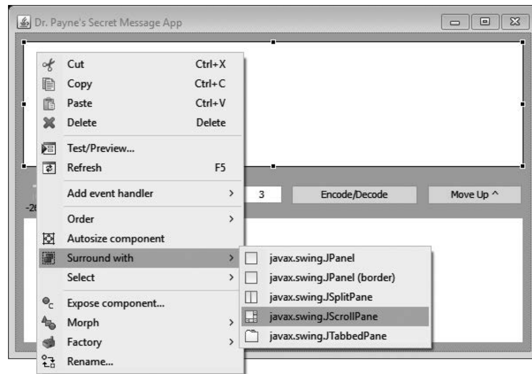


图 7-24　给 JTextArea 添加自动滚动条

你还需对另一个 JTextArea 执行同样的操作。完成这些修改后运行应用，并在输入文本区域中输入很长的文本，滚动条将自动出现。在图 7-25 中，我将整部美国宪法都粘贴到输入文本区域，并对其进行加密。

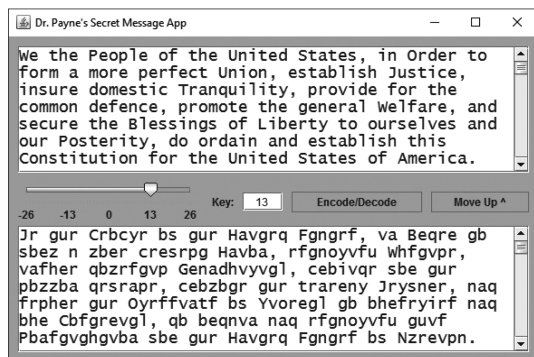


图 7-25　将每个 JTextArea 都放在 JScrollPane 内后，就可对整部美国宪法乃至更长的文本进行加密！

在图 7-25 中，只有部分文本是可见的，但两个文本区域的右边都有滚动条。看到滚动条后，我们就知道还有其他没有显示出来的文本，我们只需向下滚动就能看到它们。

为进一步定制这个应用，可在 Properties 面板中修改每个 GUI 组件的属性，如背景色、字体等。请根据喜好定制这个应用，并向朋友展示你的创意！

7.8.3　编程练习 3：在用户修改文本框内容时相应地调整滑条

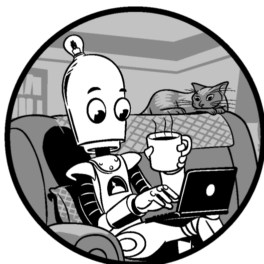
在这个编程练习中，你将对用户界面做最后一项调整。用户移动滑条时，文本框中的密钥值会相应地调整，但如果我们希望文本框中的值发生变化时，滑块也将相应地移动，该怎么做呢？

提示　你需要给文本框 txtKey 添加一个事件处理程序。为此，右击文本框 txtKey 并选择 Add event handler▶key▶keyReleased，这将创建一个事件处理程序，对文本框 txtKey 中的键击事件进行监听。

为实现前述功能，需要为这个事件处理程序编写这样的代码：获取文本框中的整数值，并将滑块设置为相应的值。处理用户提供的文本时，别忘了使用 try-catch 块。祝你好运！

第 8 章

创建移动版 Secret Messages 应用并与朋友分享



本章将创建 Secret Messages 应用的移动版，它能够通过短信或 E-mail 发送密文，还能将其发布到社交媒体。

这个应用的界面与第 7 章的 GUI 版类似：包含标签和文本框，可通过滑动调整密钥值以快速加密和解密消息。用户也可将该应用中的消息复制并粘贴到 E-mail 和短信中，但到本章结束时，用户可直接在这个应用中发送 E-mail 和短信。图 8-1 显示了这个应用在真实 Android 设备上的运行情况。

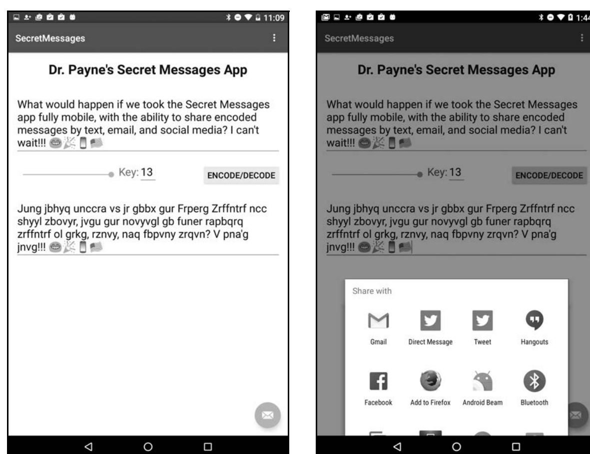


图 8-1　Secret Messages 加密/加密应用（左）；用户只需点击一个按钮就可以在这个应用中直接发送短信和 E-mail（右）

在 Android 版中，我们将像桌面版那样给 GUI 组件名称，以重用原来的代码，这都是拜 Java 语言和结构的跨平台一致性所赐。

8.1 创建移动项目

我们首先使用 Android Studio 新建一个项目。启动时，Android Studio 可能询问你是否要更新，这通常是个不错的注意。更新有一定的风险，因为更新可能移动图标或菜单项，导致程序的外观发生变化，但也可能包含重要的安全功能修复以及最新的 Android 功能。

Android Studio 刚启动时，你可能会看到最近开发的项目，这里为猜数游戏。可选择菜单 File▸Close Project 将其关闭。

为创建移动版 Secret Messages 应用，可单击欢迎屏幕中的 Start a new Android Studio project，也可在 Android Studio 中选择菜单 File▸New Project。

将项目命名为 SecretMessages，将其存储到你喜欢的位置，再单击 Next 按钮。

像本书前面创建猜数游戏一样，在 Target Android Devices 屏幕中，选择 Phone and Tablet，并将 Minimum SDK 设置为 API 16 Android 4.1（Jelly Bean），再单击 Next 按钮。在 Add an Activity 屏幕中，选择 Basic Activity，再单击 Next 按钮。在 Customize the Activity 屏幕中，保留所有默认名称不变，并单击 Finish 按钮。

配置新项目可能需要一段时间。新项目打开后，在左边的 Project Explorer 中，依次展开文件夹 app、res 和 layout，再双击其中的文件 content_main.xml。你可能需要关闭有关浮动操作按钮的弹出框（后面将用到浮动操作按钮）。

此时的屏幕应类似于图 8-2（你可能需要单击选项卡 Design 以显示 GUI 预览）。

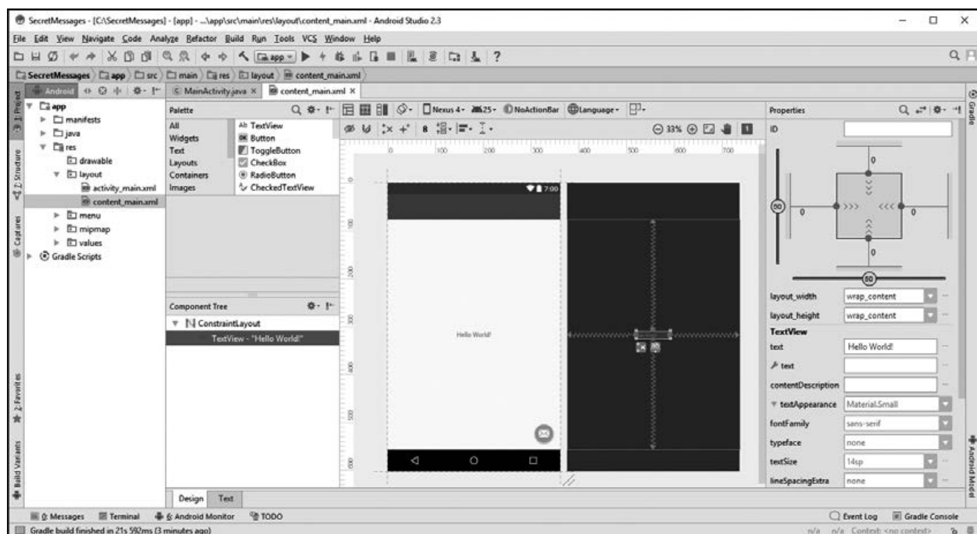


图 8-2 Android Studio 显示的应用 SecretMessages 的初始状态

8.2　设计移动 GUI

新建项目后，我们来设计应用的布局。首先，将设计预览中的文本“Hello World!”删除：单击以选择它，再按 Delete 键。然后，与前一个应用一样，添加一个用于放置 GUI 元素的 RelativeLayout：在 Palette 面板中，选择 Layouts 下的 RelativeLayout，并将其拖放到 Preview 面板或 Component Tree 中的 ConstraintLayout 上。

接下来，在应用顶部添加一个标题：在 Palette 面板中，选择 Widgets 下的控件 `TextView`，将其放在屏幕顶部中央，将属性 `text` 改为 *`Your Name`*`'s Secret Messages App`，再将其 `textAppearance` 属性改为 `Material.Large`，如图 8-3 所示。

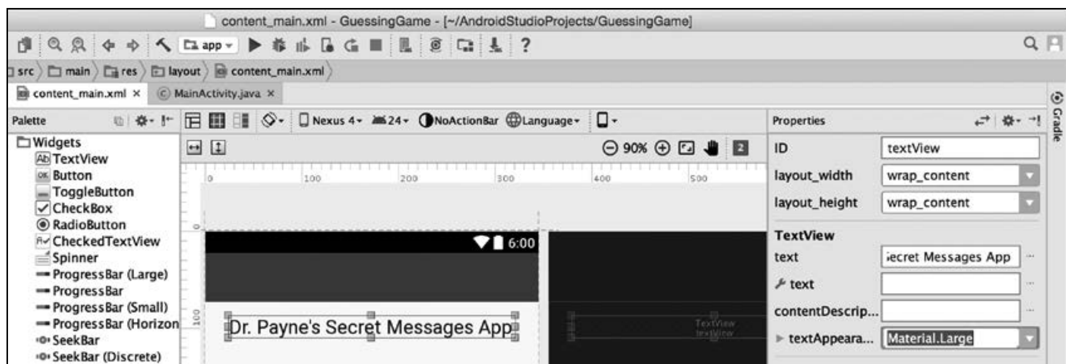


图 8-3　在应用顶部中央添加一个大型标题（请将其中的姓名改成你的）

在 Properties 面板中，展开 textAppearance 并找到 textStyle，再单击复选框 bold 让标题更突出。

下面来添加让用户输入消息的文本框。在 Palette 面板中，选择 Text 下的 Multiline Text。放大设计预览，并将文本框放在标题下方大约 30 dp 处，再将 `text` 属性改为 `Secret messages \n are so cool, \n aren't they?`，如图 8-4 所示。转义序列`\n` 会添加**换行符**；这些转义字符不会出现在屏幕上，而表示换行符（相当于在键盘上按回车键）。通过在多行中显示文本，可让用户知道他也可输入多行的消息。

还可以修改用户在特定时点可看到的文本行数。例如，可在 Properties 面板中单击链接 View all properties，再将属性 `lines` 改为 `4`。为确保在所有设备上都能看到多行消息，请单击属性 `inputType` 的下拉列表，并选择复选框 textMultiLine。

最后，将这个输入文本框的 `id` 属性改为 `txtIn`，这既有助于我们在代码中使用它，还可确保名称与 GUI 桌面版保持一致。现在，拖曳这个文本框的左右边框，将其扩大到与应用等宽，如图 8-5 所示。

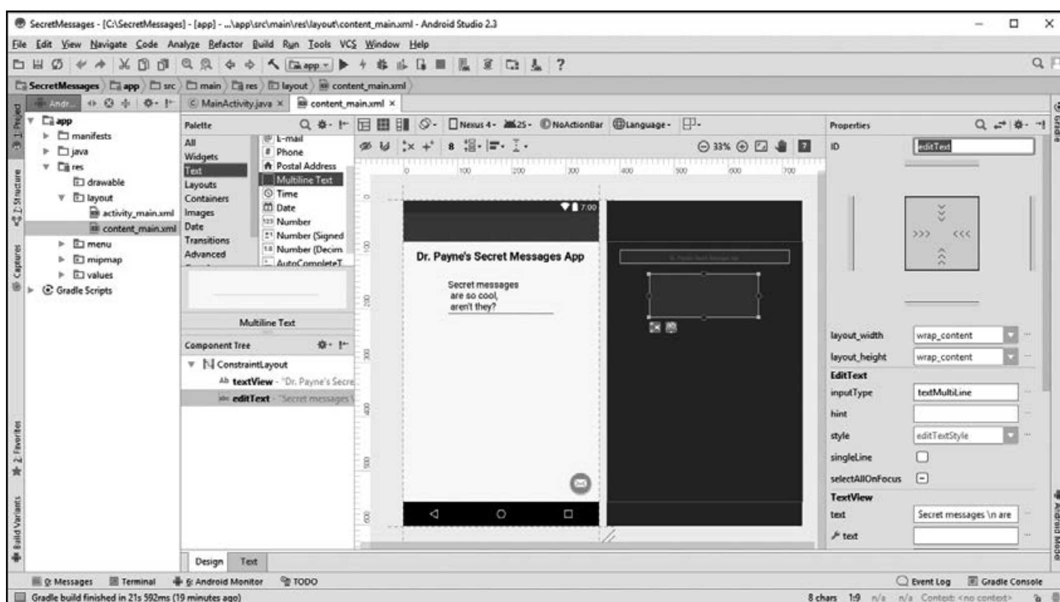


图 8-4　添加供用户输入消息的 Multiline Text，并使用转义序列\n 让文本横跨多行

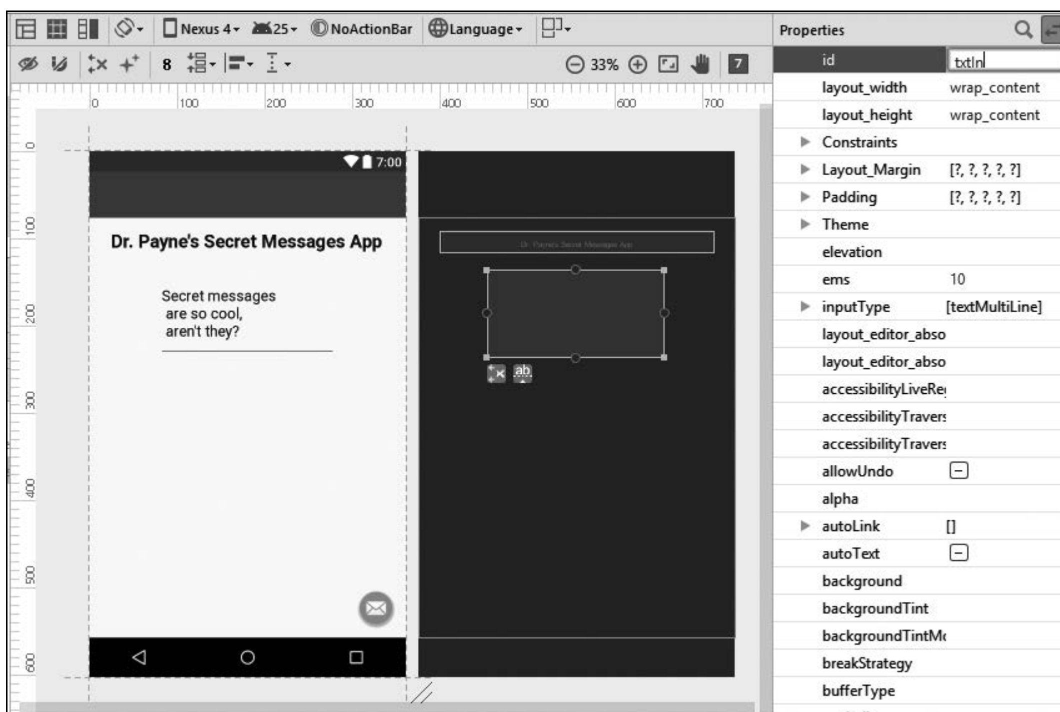


图 8-5　将这个输入文本框的 id 改为 txtIn，以方便后面给它编写代码

我们要在应用 Secret Messages 的中央添加 4 个控件：一个供用户通过滑动来指定密钥的 SeekBar；一个标签；一个显示密钥的文本框；一个 Encode/Decode 按钮。SeekBar 的作用与桌面版中的 JSlider 类似。如果这些组件没有水平对齐，也不用担心，后面将修复这个问题。

首先，在 Palette 面板的 Widgets 部分找到 SeekBar。将其放在输入文本框下方大约 30dp 处，并使其与该文本框左对齐，如图 8-6 所示。可在 Properties 面板中手动设置上外边距，为此可单击 View all properties（Properties 面板顶部的左右箭头），并将 Layout_Margin 下的属性 layout_marginTop 设置为 30dp。

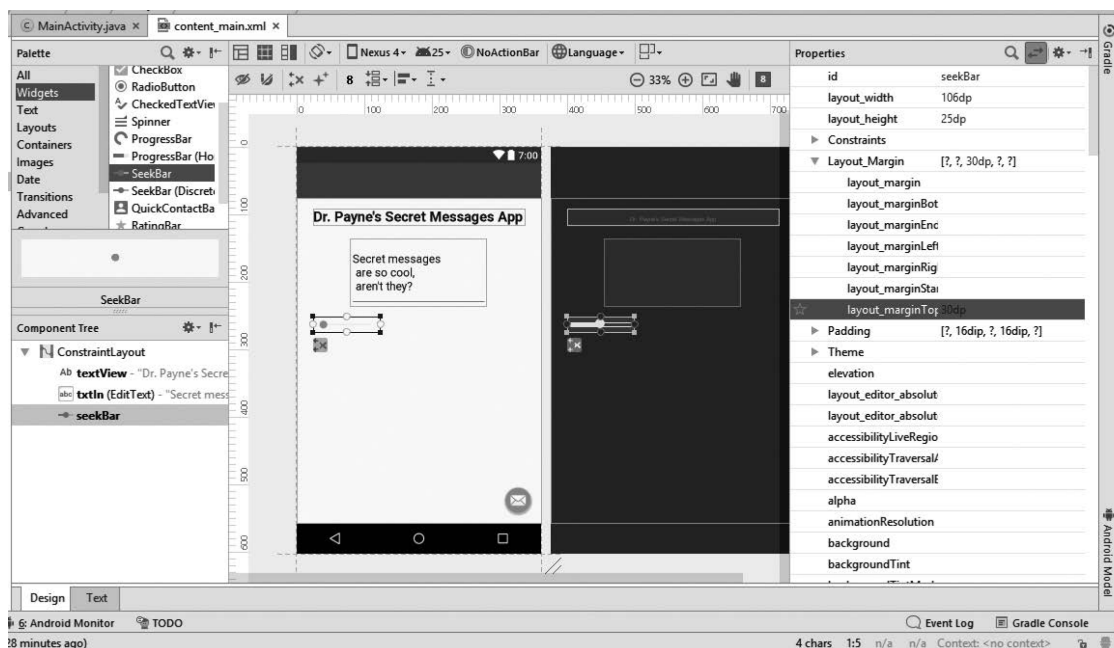


图 8-6　SeekBar 的作用与 GUI 桌面版中的滑条相同，让用户能够通过滑动指定不同的密钥值

接下来，在 SeekBar 旁边添加一个 TextView 控件，用作供用户输入密钥值的文本框的标签。在 Properties 面板中，将 textAppearance 属性改为 Material.Medium，并将 text 属性改为 Key:。

然后，添加一个供用户输入密钥的 Number 文本框，将其水平居中，并将 text 属性设置为 13 或你选择的其他默认密钥。修改这个文本框的宽度，使其刚好能够容纳数字——将 width 属性改为大约 40dp（需要先单击 View all properties），再将 id 属性改为 txtKey。

添加文本框 txtKey 后，将标签 Key:拖放到它旁边。你还应增大 SeekBar，使其填满余下的空间，如图 8-7 所示。

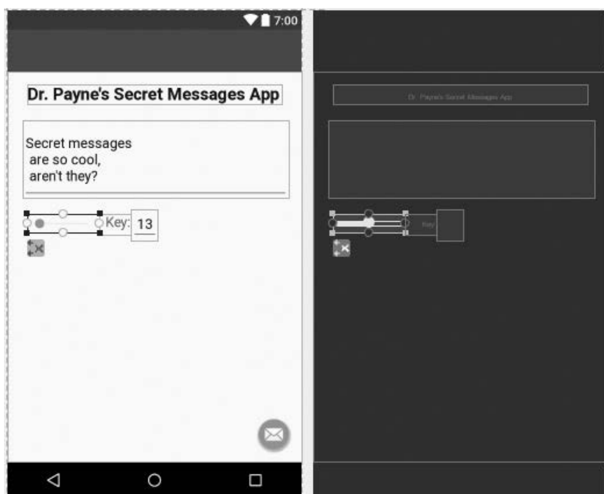


图 8-7　添加文本框 txtKey 后，调整标签 Key:的位置并增大 SeekBar

现在来添加按钮 Encode/Decode，以完成布局的中间这行。为此，在 Palette 面板中，选择 Widgets 下的 Button，将其放在文本框右边，并将 text 属性改为 Encode/Decode。

最后，创建一个显示输出消息的文本框：复制文本框 txtIn 并将其放在中间那行控件的下方。删除该文本框中的文本，将其扩大到与应用等宽，并将其 id 属性改为 txtOut。

这就好了，最终的 GUI 布局如图 8-8 所示。

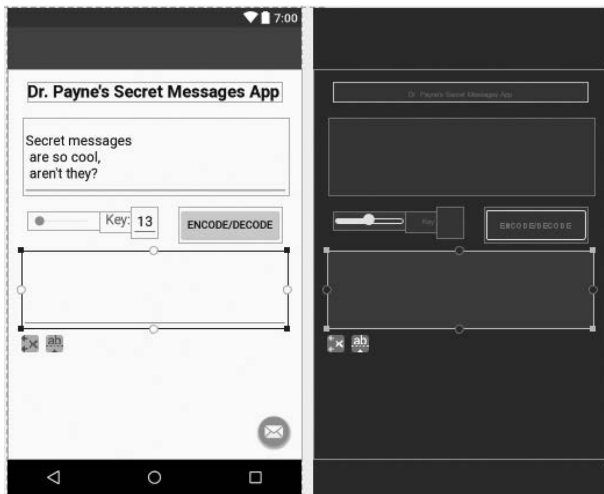


图 8-8　移动版 Secret Messages 应用的最终布局

接着往下做之前，核实输入文本框的 id 为 txtIn，输入文本框的 id 为 txtOut，中间那个文本框的 id 为 txtKey。我们必须确保这些控件的 id 都正确无误，这样下一节编写的代码才管用。

8.3　将 GUI 关联到 Java 代码

该将 GUI 布局关联到 Java 源代码，以便通过编程指定移动版 Secret Messages 应用的行为了。单击主内容窗口左上角的 MainActivity.java，切换到这个文件。

在声明 public class MainActivity 后面添加如下 5 行代码：

```
public class MainActivity extends AppCompatActivity {
    EditText txtIn;
    EditText txtKey;
    EditText txtOut;
    SeekBar sb;
    Button btn;
```

添加每行代码后，你可能需要按 Alt-回车键或 Option-回车键导入相应的类。

这 5 个变量将用来表示布局中的 GUI 组件。为将这些变量关联到实际控件，可在 onCreate()方法中（语句 setSupportActionBar(toolbar);后面）添加如下 5 行代码：

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    txtIn = (EditText)findViewById(R.id.txtIn);
    txtKey = (EditText)findViewById(R.id.txtKey);
    txtOut = (EditText)findViewById(R.id.txtOut);
    sb = (SeekBar)findViewById(R.id.seekBar);
    btn = (Button)findViewById(R.id.button);
```

对于这里的大多数语句，都可依赖 Android Studio 的代码助手来自动补全：你只需输入前几个字符，再单击列表项或按回车接受代码建议。这样做速度更快，且有助于避免常见的输入错误。

8.3.1　将按钮 Encode/Decode 关联到方法 encode()

将代码与 GUI 布局中的组件相关联后，就可复制第 7 章创建的桌面版中的方法 encode()了。我们将把这个方法粘贴到 MainActivity 类声明中——放在刚添加的 5 行代码（它们声明了指向布局中 GUI 组件的变量）后面。

在 Eclipse 中，打开桌面项目 SecretMessagesGUI，并选择整个 encode()方法。复制这些代码，并将其粘贴到 Android Studio 中的 MainActivity 类声明中：

```
public class MainActivity extends AppCompatActivity {
    EditText txtIn;
    EditText txtKey;
    EditText txtOut;
    SeekBar sb;
    Button btn;
    public String encode( String message, int keyVal ) {
```

```
        String output = "";
        char key = (char) keyVal;
        for ( int x = 0; x < message.length(); x++ ) {
            char input = message.charAt(x);
            if (input >= 'A' && input <= 'Z')
            {
                input += key;
                if (input > 'Z')
                    input -= 26;
                if (input < 'A')
                    input += 26;
            }
            else if (input >= 'a' && input <= 'z')
            {
                input += key;
                if (input > 'z')
                    input -= 26;
                if (input < 'a')
                    input += 26;
            }
            else if (input >= '0' && input <= '9')
            {
                input += (keyVal % 10);
                if (input > '9')
                    input -= 10;
                if (input < '0')
                    input += 10;
            }
            output += input;
        }
        return output;
    }
```

方法 encode()就绪后，我们只需在按钮 Encode/Decode 被单击时调用它。为此，我们将为 Encode/Decode 按钮创建 OnClickListener，并在其中调用方法 encode()。

在 MainActivity.java 的方法 onCreate()中，输入代码 btn.setOnCl，将出现一个代码推荐列表，如图 8-9 所示。请选择其中的 setOnClickListener()。

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    txtIn = (EditText)findViewById(R.id.txtIn);
    txtKey = (EditText)findViewById(R.id.txtKey);
    txtOut = (EditText)findViewById(R.id.txtOut);
    sb = (SeekBar)findViewById(R.id.seekBar);
    btn = (Button)findViewById(R.id.button);

    btn.setOnCl
        setOnClickListener(OnClickListener l)
        setOnContextClickListener(OnC...    void
        setOnLongClickListener(OnLong...    void
```

图 8-9　可依赖 Android Studio 的代码推荐功能自动补全代码，如这里显示的 Encode/Decode 按钮的 setOnClickListener()

单击代码推荐 setOnClickListener()后，在这个方法的括号内单击，并输入 new OnC，再双击第一个代码推荐，如图 8-10 所示。

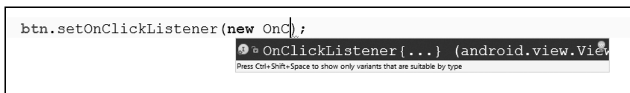


图 8-10　再次使用代码助手来补全 OnClickListener；这次 Android Studio 提供了多行代码

你将发现 Android Studio 给事件处理程序 OnClickListener()添加了多行代码：

```
btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
    }
});
```

这几行代码使用匿名内部类 View.OnClickListener()创建了一个事件监听器。这个匿名内部类包含一个 onClick()事件处理程序，在用户单击该按钮时做出响应。

接下来，为调用方法 encode()做好准备。encode()不关心其他代码中发生的情况——只要我们给它提供两个数据类型正确的参数。这意味着我们无需修改 encode()，而只需准备好两个用作参数的值。

为对用户在 txtIn 中输入的消息进行加密，首先需要获取用户在 txtKey 中输入的密钥值，再获取用户在 txtIn 中输入的文本字符串。我们将把这两个值作为参数传递给方法 encode()，并将加密结果存储在变量 output 中。最后，将把文本框 txtOut 的 text 属性设置为加密得到的输出字符串（变量 output）。

请在方法 onClick()的大括号内添加如下 4 行代码：

```
btn.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
     ❶ int key = Integer.parseInt(txtKey.getText().toString());
     ❷ String message = txtIn.getText().toString();
     ❸ String output = encode(message, key);
     ❹ txtOut.setText(output);
    }
});
```

在❶处，我们使用方法 Integer.parseInt()将用户输入的文本转换为整数值，并将其存储到变量 key 中。与以前相比，这里使用 parseInt()时唯一不同的是，txtKey 后面有两个方法：getText()和 toString()。在 Android 中，EditText（文本框）的方法 getText()返回的不是字符串，而是一种名为 Editable 的灵活类型。所有 EditText 类型（从 Plain 到 Password 再到 Number）都返回一个 Editable 对象，因为 EditText 中的文本是供用户修改和编辑的。我们使用方法 toString()将 EditText 返回的文本转换为字符串，以便能够通过分析找出用户输入的数字。

❷处的情况也是如此：我们需要使用方法 getText()和 toString()从 txtIn 获取输入消息，并将其转换为可进行加密的字符串。在❸处，我们调用方法 encode()并将用户提供的消息和密钥传递给它，以便对消息进行加密；然后将结果存储在变量 output 中。最后，在❹处，我们设置文本框 txtOut 的 text 属性，以显示加密得到的输出（变量 output）。

至此，这个应用的功能已足够多，可进行第一次测试了。我们在 Android 模拟器中运行这个应用，看看 Encode/Decode 按钮是否管用。

8.3.2　测试应用

将所做的工作存盘，并选择菜单 Run ▸ Run 'app'。选择第 4 章创建的模拟器，如 My Nexus 6P，如图 8-11 所示。

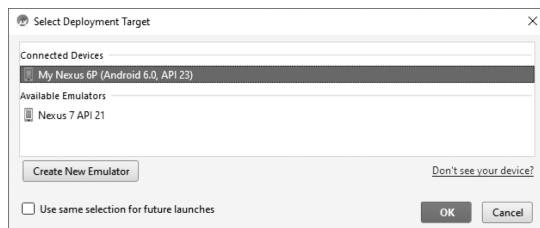


图 8-11　选择模拟器

模拟器可能需要几分钟才能启动——别忘了你可以让它一直运行，以免运行应用时需要很长的启动时间。你也可像第 4 章那样跳过模拟器，直接在 Android 设备上运行这个应用。

模拟器启动并加载应用 Secret Messages 后，你将看到初始布局，如图 8-12（左）所示。输入密钥并单击 Encode/Decode 按钮，你将看到 Android 将消息转换成了密文，如图 8-12（右）所示。

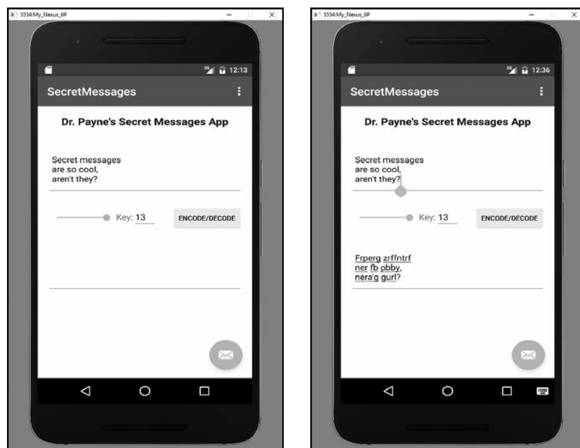


图 8-12　在 Android 模拟器 Nexus 6P 中运行的应用 Secret Messages（左）；单击 Encode/Decode 按钮生成密文（右）

你可在两个文本框之间复制并粘贴，但滑条/SeekBar 目前还不管用，浮动操作按钮（右下角的信封符号）也没有任何作用。请不要担心，我们马上就会处理这些问题。

注意，还存在另一个用户体验方面的问题：密文（Frperg zrffntrf...）带下划线，因为它们在计算机看来拼写不正确。为关闭拼写检查，请切换到布局文件 content_main.xml，并选择文本框 txtOut，再在 Properties 面板中单击属性 inputType 的值并选择复选框 textNoSuggestions。如果你愿意，也可对文本框 txtIn 关闭拼写检查；为此，可分别设置这两个文本框，也可同时选择这两个文本框（按住 Shift 键并单击），再选择复选框 textNoSuggestions。再次在模拟器中运行应用，现在不再执行拼写检查了。

完成这项细微的调整后，就可给 SeekBar 编写代码，让用户能够轻松、快捷地调整密钥。

8.3.3　给 SeekBar 编写代码

现在来给 SeekBar 编写代码，使得用户左右滑动时相应地修改输出消息。为此，需要修改 SeekBar 的几个属性，为使用它做好准备。

先简单地介绍一下 SeekBar 控件。你在移动设备上播放视频时可能见过 SeekBar，例如，播放 YouTube 视频时，SeekBar 显示播放到了什么地方，并让你能够快进或快退——这有时被称为**定位**（seeking）。

相比于 JSlider，SeekBar 有几个不同之处。首先，SeekBar 的取值不能为负，其最小值为 0，但你可将最大值设置为任何正数，如 26。另外，SeekBar 不会显示刻度和标签。这意味着要让用户能够使用负的密钥值，必须通过数学运算将正值转换为负值，让用户能够在包含负数的区间（如–13~+13）内选择密钥。为此，我们将把 SeekBar 的取值范围设置为 0~26，并在用户移动 SeekBar 时更新文本框 txtKey 显示的值，让用户知道 SeekBar 的当前值。下面首先来修改 SeekBar 的一些属性。

切换到布局文件 content_main.xml，并选择设计预览中的 SeekBar。在 Properties 面板中，找到属性 max 和 progress，并将它们的值都改为 26。

属性 max 指定了 SeekBar 的最大可能取值。就应用 Secret Messages 而言，至少需要 26 个值才能覆盖整个字母表。我们希望范围为–13~+13，这样用户可轻松地加密和解密。在这个范围内，总共包含 27 个值，因此我们将 max 属性设置为 26（这将包含 27 个值——如果将 0 也算在内）。属性 progress 表示 SeekBar 的当前值，它类似于 JSlider 的属性 value。我们将使用 SeekBar 的方法 getProgress()来获取 SeekBar 的当前值。

保存所做的修改，再切换到源代码文件 MainActivity.java。在方法 onCreate()中 btn.setOnClickListener()的最后一行代码});后面，输入 sb.set 并使用代码推荐器找到 setOnSeekBarChangeListener()。

SeekBar 监听器的代码与 Encode/Decode 按钮的监听器不同。我们要监听的不仅是 SeekBar 单击事件，而要监听所有变化，包括用户按住 SeekBar 并左右滑动。为此，我们需要使用 OnSeekBarChangeListener。这会将 SeekBar 存储在变量 sb.中。

然而，与给按钮编写监听器一样，我们将使用代码推荐器来自动生成新的 OnSeekBarChange

Listener。为此，在 sb.setOnSeekBarChangeListener()的括号内，输入 OnSeekBar 并使用代码推荐。

Android Studio 将在 OnSeekBarChangeListener 中添加三个方法：onProgressChanged()、onStartTrackingTouch()和 onStopTrackingTouch()。我们要使用的是 onProgressChanged()，因为它在用户修改了 SeekBar 的值时被调用。

在 SeekBar 的事件监听器中，我们要执行的操作与 Encode/Decode 按钮的事件监听器中类似，但需要根据 SeekBar 的值计算密钥值。我们还需在文本框 txtKey 中显示密钥。请在方法 onProgressChanged()中添加如下代码行：

```
sb.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean
fromUser) {
      ❶int key = sb.getProgress() - 13;
        String message = txtIn.getText().toString();
        String output = encode(message, key);
        txtOut.setText(output);
      ❷txtKey.setText("" + key);
}
```

在❶处，我们创建一个名为 key 的变量。我们从 SeekBar 变量 sb 那里获取当前值，并将其减去 13。SeekBar 的可能取值为 0~26，因此减去 13 后的取值范围为–13~+13，这非常完美，让用户能够加密和解密消息。接下来的 3 行与 Encode/Decode 按钮的事件处理程序代码完全相同，它们获取消息、使用方法 encode()对消息加密并在文本框 txtOut 中显示密文。

❷处的代码行是新增的。我们要在用户移动 SeekBar 时相应地修改文本框 txtKey 中的值。由于 key 是一个整型变量，我们将一个空字符串（""）与它相加，以得到一个文本字符串，并将其显示给用户。你也可以使用 Integer 类的方法 toString()（即使用代码 Integer.toString(key)）将 key 转换为字符串，但将空字符串与 int 相加是一种方便的快捷方式。

8.4　在模拟器和 Android 设备上运行应用

保存所做的修改，再尝试在模拟器中运行应用。你可左右滑动 SeekBar，密文将相应地变化，如图 8-13 所示。

鉴于连接 Android 设备还是几章前的事，我们来简要地复习一下。首先，使用兼容的 USB 电缆将 Android 设备连接到计算机，出现询问你是否要启动 USB 调试的对话框时，单击 Yes。

在 Android Studio 中，单击 Run 按钮，将出现 Select Deployment Target 对话框（你可能需要在模拟器中关闭应用，这样才会出现这个对话框）。选择你连接的 Android 设备，再单击 OK 按钮。

在 Android 设备上运行这个应用时，你将发现 SeekBar 的响应速度更快。另外，你还可以点击应用的某些地方来访问一些选项。如果你点击密文并选择 Select All，将出现一个上下文菜单，其中包含菜单项 Cut、Copy、Share 和 Assist，如图 8-14 所示。



图 8-13　现在 SeekBar 功能齐备了，这里显示的是在 Android 模拟器中的情况，其中密钥值为 5

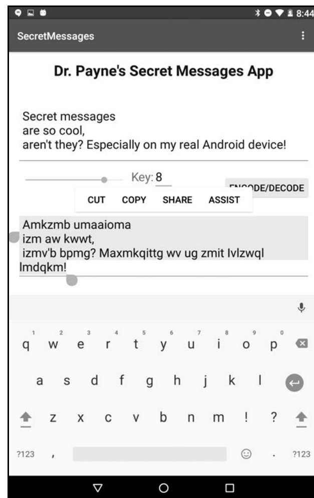


图 8-14　上下文菜单让你能够剪切、复制或分享密文

如果你点击 Share，就可通过 E-mail、短信、聊天程序、蓝牙、Twitter、Facebook 或其他支持分享的应用来发送选定的文本，如图 8-15（左）所示。通过使用菜单项 Share，你可直接在应用中将密文发送给任何人。你还可通过 E-mail 将密文发送给安装了桌面 GUI 版 Secret Messages 应用的朋友，如图 8-15（右）所示。收到邮件后，朋友只需将其中的密文复制到桌面 GUI 版 Secret Messages 应用中，并使用相反的密钥进行解密。例如，对于使用密钥 8 加密的消息，可使用密钥–8 进行解密。

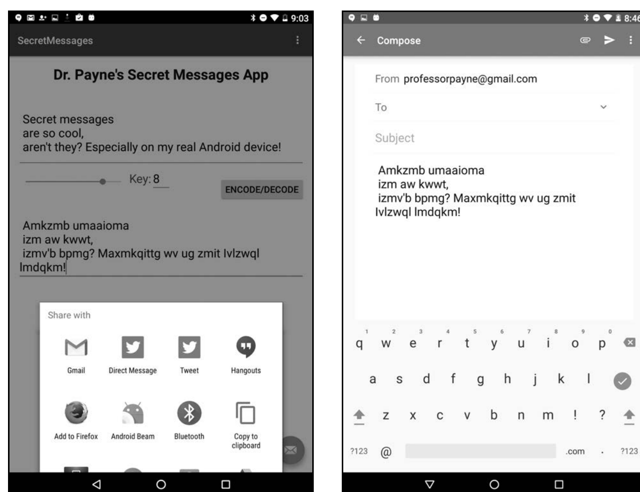


图 8-15　菜单项 Share 让你能够通过 E-mail、短信、Twitter、Facebook 或其他支持分享的应用分享密文（左）；如果你选择选项 E-mail，将自动创建一封包含密文的邮件（右）

至此，你已创建了功能齐备的移动版 Secret Messages 应用，让用户能够通过 E-mail、短信或社交媒体与任何人分享密文。

为了让这个应用的功能更为完备，我们将给浮动操作按钮添加自定义操作，让用户只需用手指点一下就能分享密文。

8.5　定制浮动操作按钮

前面我们一直对浮动操作按钮（也叫 fab 图标）置若罔闻，这是应用屏幕右下角的圆形信封符号，如图 8-16 所示。

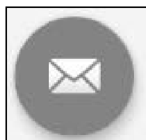


图 8-16　浮动操作按钮

fab 图标让我们能够向用户提供执行特定任务（如通过 E-mail、短信或社交媒体分享消息）的快捷途径。

Android Studio 在方法 onCreate()中自动给 fab 图标添加了一些代码，这些代码显示一种弹出消息——在 Android lingo 中称为 Snackbar：

```
FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
fab.setOnClickListener(new View.OnClickListener() {
    @Override
```

```
    public void onClick(View view) {
        Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                .setAction("Action", null).show();
    }
});
```

我们只需用所需代码替换 fab.setOnClickListener()的方法 onClick()的代码。

首先，将方法 onClick()的大括号内的代码删除。然后，在方法 onClick()中输入如下代码：

```
FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
     ❶ Intent shareIntent = new Intent(Intent.ACTION_SEND);
     ❷ shareIntent.setType("text/plain");
     ❸ shareIntent.putExtra(Intent.EXTRA_SUBJECT, "Secret Message "+
                DateFormat.getDateTimeInstance().format(new Date()));
     ❹ shareIntent.putExtra(Intent.EXTRA_TEXT, txtOut.getText().toString());
     ❺ try {
            startActivity(Intent.createChooser(shareIntent, "Share message..."));
            finish();
        }
     ❻ catch (android.content.ActivityNotFoundException ex) {
            Toast.makeText(MainActivity.this, "Error: Couldn't share.",
                    Toast.LENGTH_SHORT).show();
        }
    }
});
```

我们来看看这些代码都是做什么的。在❶处，我们创建一个名为 shareIntent 的 Intent。在 Android 中，Intent 是我们要启动的活动，如 E-mail、Twitter 或相机。shareIntent 使用了 ACTION_SEND，这意味着我们要向 Android 设备中的另一个应用发送信息。

我们将 shareIntent 的类型设置成 text/plain❷——E-mail、推文和社交媒体帖子的类型，再给活动添加一个主题行❸。如果用户选择通过 E-mail 或其他使用主题行的应用分享消息，主题行将为 Secret Message 和当前日期。DateFormat 类根据用户所在的地区将日期格式化为文本形式，如"Dec 7, 2017 4:33:12 PM"。

❹处的代码行根据用户选择用来分享消息的应用，将文本框 txtOut 中的密文作为 E-mail、推文或帖子的正文。

接下来是一条 try 语句❺。在一个应用中调用另一个应用时，很多地方都可能出错。如果用户没有安装 E-mail 应用，结果将如何呢？如果应用正忙或网络不可用呢？这条 try 语句就是为这样的情形准备的。在这条 try 语句中，第 1 行代码尝试启动用户选择的活动（如 E-mail、短信或 Twitter），再将信息传递给选定的应用，而第 2 行代码结束这个活动。

如果出现错误或异常，catch 语句❻将显示一条内容为"Error Couldn't share."的 Toast 消息（弹出框）。

这就是让 fab 图标发挥作用所需做的全部工作。输入这些代码行时，别忘了按 Alt-回车键（或

Option-回车键）导入相关的类。别忘了，如果 Android Studio 在类名下方显示了红线，通常是由于这个类未导入，因此像第 4 章所做的那样按 Alt-回车键（或 Option-回车键）将自动添加相应的 `import` 语句。

添加代码并导入必要的类后，再次运行这个应用。

将消息加密后，点击屏幕底部向下的三角形将键盘隐藏，再点击屏幕右下角的 fab 图标，如图 8-17（左）所示。从 Share message...下方的应用列表中选择你的 E-mail 应用，该应用将启动，其中的邮件包含主题行，正文为密文，如图 8-17（右）所示。

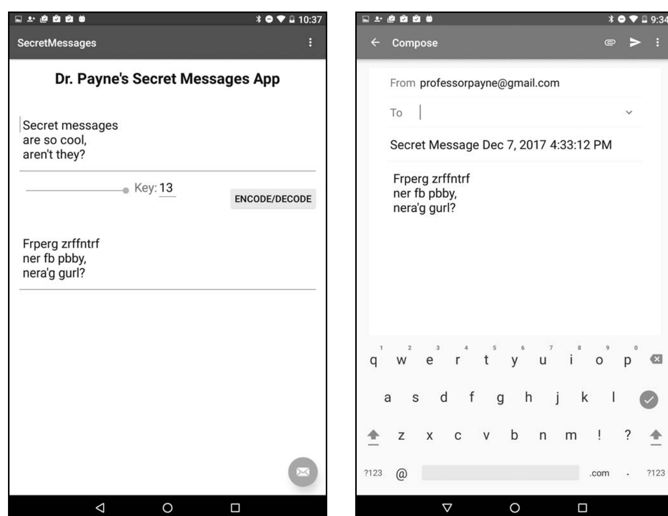


图 8-17 点击右下角的 fab 图标将弹出一个分享菜单（左）；从分享菜单中选择 E-mail 应用将自动创建一封邮件，其正文为密文，而主题行为当前日期和时间（右）

现在，用户只需点击 fab 图标就能在应用中发送密文了。但为吧让这个应用有资格进入 Google Play Store，我们还需让它能够接收来自其他应用的信息。

8.6 接收来自其他应用的信息

下面对移动版 Secret Messages 应用做最后一项改进。在这个应用中，可通过 E-mail 或 Twitter 分享密文，但如果要接收来自 E-mail、推文和帖子中的文本，以免复制并粘贴它们，该如何做呢？为解决这个问题，可将应用 Secret Messages 添加到 Android 分享列表中，这样其他应用就能直接向它发送文本了，而这只需几行代码就能实现。

这是一种高阶技术，这里将详细介绍其步骤。首先，在 Project Explorer 面板中，打开文件夹 manifests 中的文件 AndroidManifest.xml。这个清单文件存储了应用的“货运”信息，就像货轮的载货单列出了其运载的所有货物一样。你可在其中声明应用的属性，还可包含运行应用时 Android 设备需要知道的详细信息，就像你发送越洋包裹时可标注其中的货物一样。

为此，需要编辑一些 XML。XML 文件包含**标签**，以对文件中的信息进行标记。通常，标签由名称和包含名称的尖括号组成，如<tag>。在这里，我们需要修改清单文件中的标签 intent-filter，其中包含有关应用将接受哪些数据的信息。请在文件 AndroidManifest.xml 的末尾附近（标签</activity>前面）添加如下 5 行 XML 代码：

```
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        ❶ <intent-filter>
            ❷ <action android:name="android.intent.action.SEND" />
            ❸ <category android:name="android.intent.category.DEFAULT" />
            ❹ <data android:mimeType="text/*" />
        ❺ </intent-filter>
        </activity>
    </application>
</manifest>
```

在❶处，我们在应用中添加了一个 intent-filter。前面我们使用了一个 Intent 对象来向其他应用（如 E-mail、Twitter 或 Facebook）发送文本，而在这里，我们创建了一个过滤器，以接收当前 Android 设备上其他应用发送的 Intent 对象❷（在 fab 图标的代码中，我们使用 Intent.ACTION_SEND 向其他应用发送 Intent 消息，而这里所做的与此相反）。

❸处的代码行使用分享类别 DEFAULT 将应用 Secret Messages 添加到可与之分享的应用列表中。使用分享类别 DEFAULT 时，应用可接收来自任何应用的数据（还有其他的分享类别，可用来过滤或限制可向你的应用发送数据的应用类型。例如，对于只想接收来自 Web 浏览器的数据的应用，可使用类别 BROWSABLE）。在❹处，我们告诉 Android，应用 Secret Messages 可接收来自其他应用的任何文本数据。（如果你编写的应用将图像作为输入，可将 mimeType 设置为"image/*"或"image/png"（仅 PNG 图像）；对于将视频作为输入的应用，可将 mimeType 设置为"video/mp4"。）

最后，我们结束标签 intent-filter❺。XML 标签必须结束，这通常是使用斜杠和 XML 标签名实现的，如</intent-filter>。这让计算机知道 intent-filter 的代码到此结束。将文件 AndroidManifest.xml 存盘，再打开文件 MainActivity.java 以编辑 Java 源代码。

在文件 MainActivity.java 中向下滚动，以找到方法 onCreate()。在将 GUI 组件关联到变量 txtIn 到 btn 的代码后面，添加如下 4 条语句：

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    txtIn = (EditText)findViewById(R.id.txtIn);
    txtKey = (EditText)findViewById(R.id.txtKey);
    txtOut = (EditText)findViewById(R.id.txtOut);
    sb = (SeekBar)findViewById(R.id.seekBar);
    btn = (Button)findViewById(R.id.button);
```

```
❶ Intent receivedIntent = getIntent();
❷ String receivedText = receivedIntent.getStringExtra(Intent.EXTRA_TEXT);
❸ if (receivedText != null)
    ❹ txtIn.setText(receivedText);
   btn.setOnClickListener(new View.OnClickListener() {
```

在❶处，我们创建了一个名为 receivedIntent 的变量，用于接受其他应用分享的 Intent 消息。然后，我们调用方法 getStringExtra()获取 Intent 消息中的文本❷。Intent 发送的消息的组成部分被称为额外（extra）数据。在❸中，使用了一条 if 语句来检查变量 receivedText 是否不为 null。如果确实不为 null，就将应用 Secret Messages 顶部的文本框 txtIn 中的文本修改为收到的文本❹。

信不信由你，这就是在应用 Secret Messages 中为接收其他应用分享的消息所需做的全部工作。现在，你不仅能够在这个应用中发送密文，还可以直接在其中解密 E-mail、推文和帖子中的信息。图 8-18 显示了将 E-mail 中的信息分享到应用 Secret Messages，以便快速对其进行解密的过程。

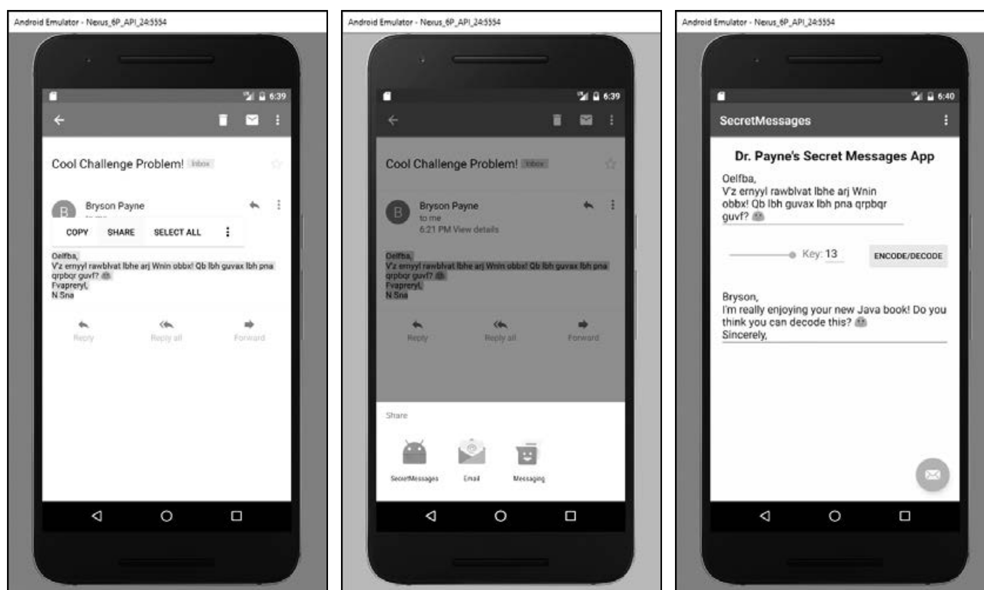


图 8-18　现在应用 Secret Messages 能够向 Android 设备中的其他应用（包括 E-mail）发送消息，还可接收来自这些应用的消息

请在你喜欢的 E-mail 应用、Twitter 或 Facebook 中选择文本，再点击 Share（分享）按钮，应用 Secret Messages 将出现在分享应用列表中。选择 Secret Messages（将其作为要与之分享消息的应用），选定的文本将出现在应用 Secret Messages 的输入文本框中。你可轻松地加密或解密，只是别忘了你可能需要根据加密时使用的密钥相应地修改密钥值。

祝贺你创建了一个完备的社交移动应用——Secret Messages！好好把玩并与朋友分享吧！

下面来看看你在本章学到的技能并尝试完成两个编程练习。

8.7 小结

在本章开发的应用中，我们重用了 GUI 桌面版 Secret Messages 应用的代码，还添加了大量代码，以使其成为一个完备的移动应用。下面是本章介绍的一些新知识以及强化的一些既有技能：

- 给包括按钮、SeekBar 和浮动操作按钮在内的 Android 组件编写事件处理程序；
- 定制 Android GUI 组件；
- 在 Android 应用中使用多行文本输入；
- 添加 SeekBar 并像滑块一样使用它；
- 在模拟器和实际设备中运行 Android 应用；
- 定制浮动操作按钮的行为；
- 创建自定义 Intent 对象，以便在应用中调用特殊活动，其中包括向其他应用发送数据；
- 创建自定义 Intent 过滤器，以便接收其他应用发送的文本（图像或视频）数据。

8.8 编程练习

为复习并使用学到的知识以及获得更多的编程技能，请尝试完成这里的编程练习。如果遇到困难，可从本书的配套网站（https://www.nostarch.com/learnjava/）下载示例解决方案。

8.8.1 编程练习 1：添加 Move Up ^按钮

在这个编程练习中，你将创建一个 Move Up ^按钮，它类似于你在第 7 章的编程练习 1 中创建的按钮，将输出文本框中的文本移到输入文本框中。

将按钮 Move Up ^放在文本框 txtOut 下方，再添加一个事件处理程序，它获取文本框 txtOut 中的文本，并将文本框 txtIn 的 text 属性设置为这些文本。另外，你还可以让这个事件处理程序将 SeekBar 的值取负；例如，如果加密密钥为 7，就将其改为–7，以用作解密密钥。最终的结果是，用户单击按钮 Move Up ^时，将把密文移到输入文本框中，并自动对其进行解密。

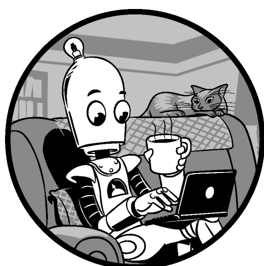
8.8.2 编程练习 2：修改 SeekBar 的属性 progress

在用户体验方面，可做的第二个改进是，当用户在文本框 txtKey 中输入密钥时，相应地修改 SeekBar 的属性 progress。请试一试这样做！

> **提示** 不要给文本框 txtKey 创建事件处理程序，而是修改按钮 Encode/Decode 的事件处理程序的代码，将 SeekBar 的属性 progress 改为文本框 txtKey 中的值与 13 的和——别忘了 SeekBar 的取值范围为 0~26，而不是–13~+13。你应该只需编写一两行代码。

第 9 章

使用鼠标绘制五颜六色的气泡



在接下来的三章中，我们将创建交互式动画绘图应用 BubbleDraw，让用户能够使用鼠标（桌面版）和手指（移动版）绘制出漂浮而有弹性的多彩气泡。

应用 BubbleDraw 的第一个版本如图 9-1 所示，其中每个气泡的颜色都是随机的。用户在应用窗口中单击并拖曳鼠标时，将绘制出颜色随机的气泡。用户还可以通过上下滚动鼠标滑轮或者在触控板或触摸屏上使用滚动手势来调整气泡的大小。

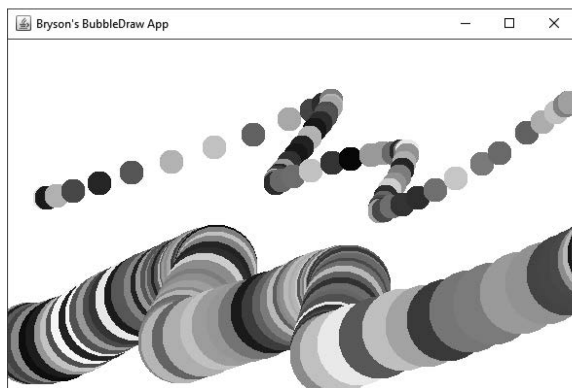


图 9-1　应用 BubbleDraw 让用户能够使用鼠标来绘制颜色随机的气泡

我们将使用面向对象编程方法来创建应用 BubbleDraw。本书前面介绍的变量、函数、循环和条件语句都属于**过程性编程**。过程性编程以线性方式循序渐进地编写程序，有点像按菜谱做菜；面向对象编程使用了前面介绍的所有概念，但将大型软件项目分成被称为**对象**的小块，让你能够开发更大且复杂得多的程序。

例如，气泡是应用 BubbleDraw 中的重要实体，因此着手编写这个应用时，就可将气泡视为对象。首先，我们将处理如何定义气泡的问题。接下来，确定如何存储有关大量气泡的信息以及如何在屏幕上绘制气泡。最后，添加通过单击并拖曳鼠标来绘制气泡的功能。我们不分别编写控制各个气泡的代码，而是编写适用于所有气泡对象的统一代码。

首先，编写两个源代码文件：表示应用窗口的 BubbleDraw 以及表示画布的 BubblePanel。应用窗口将扩展你熟悉的类 `JFrame`，而窗口内的画布将使用前面没有介绍过的 GUI 容器 `JPanel`。通过创建两个独立的文件，我们能够在第 10 章的 GUI 应用中重用画布。现在就来编写程序吧！

9.1　创建项目 BubbleDraw

在 Eclipse 中，选择菜单 File▸New▸Java Project，为 BubbleDraw 应用创建一个新的项目文件夹。将项目命名为 BubbleDraw 并单击 Finish 按钮。

在 Project Explorer 面板中，展开项目文件夹 BubbleDraw，右击文件夹 src 并选择 New▸Class。将这个类命名为 `BubbleDraw`（用于表示应用窗口），将超类设置为 `javax.swing.JFrame`，在 Which method stubs…部分选择复选框 public static void main (String[] args)，再单击 Finish 按钮。

接下来，创建表示画布的 `BubblePanel` 类。右击文件夹 src 并选择 New▸Class。将这个类命名为 `BubblePanel`，并将超类设置为 `javax.swing.JPanel`，再单击 Finish 按钮。

在这些类中，`BubblePanel` 表示画布，可在其他应用中重用并扩展它，而 `BubbleDraw` 表示窗口，是显示 `BubblePanel` 的容器。

9.2　创建框架 `BubbleDraw`

我们先在源代码文件 BubbleDraw.java 中搭建主应用窗口。在 Eclipse 中，单击内容面板顶部的选项卡 BubbleDraw.java，你将看到这个文件包含如下代码：

```
import javax.swing.JFrame;
public class BubbleDraw extends JFrame {
    public static void main(String[] args) {
    }
}
```

与前面的 GUI 应用一样，应用窗口位于一个 `JFrame` 内。在这个应用中，我们要在窗口中显示画布——`BubblePanel`。在这个应用版本中，我们不会添加其他 GUI 组件。

与前几章一样，我们要创建一个 `JFrame` 并添加设置它的代码，但我们还需将表示画布的 `BubblePanel` 添加到这个框架中。`BubbleDraw` 类的完整代码如代码清单 9-1 所示。

代码清单 9-1　`BubbleDraw` 类

```
import javax.swing.JFrame;
public class BubbleDraw extends JFrame {
    public static void main(String[] args) {
```

```
  ❶ JFrame frame = new JFrame("Your Name's BubbleDraw App");
  ❷ frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  ❸ frame.getContentPane().add(new BubblePanel());
  ❹ frame.setSize(new java.awt.Dimension(600,400));
  ❺ frame.setVisible(true);
  }
}
```

首先，创建了一个包含标题栏的 JFrame 对象❶，并将你的姓名包含在标题字符串中。接下来，设置了默认关闭操作，使得在用户关闭这个窗口时退出应用❷。然后，新建了一个表示画布的 BubblePanel 对象，并将其添加到框架中❸。最后两行代码设置了窗口的尺寸❹并让窗口可见❺。

保存文件 BubbleDraw.java 并运行应用，你将看到一个灰色 Java 框架，其标题栏中显示 *Your Name*'s BubbleDraw App。接下来给表示画布的 BubblePanel 类中添加逻辑。

9.3 创建表示气泡的类

切换到文件 BubblePanel.java 所在的选项卡，这个文件将包含所有在屏幕上绘制气泡的逻辑。我们的第一项任务是创建一个 Bubble 类，用于存储每个气泡的颜色、尺寸以及在屏幕上的位置。

9.3.1 定义气泡

创建类的原因非常现实，因为如果采用过程性编程方法，将需要使用不同的变量来存储气泡的 x 坐标、y 坐标、尺寸等。例如，对于第一个气泡，需要使用变量 bubble1x 来存储其 x 坐标，还需使用变量 bubble1y、bubble1size、bubble1color 等。如果只有一个气泡，情况不会太糟，但如果用户拖曳鼠标数秒钟，创建出 1000 个气泡呢？要跟踪这些气泡，将需要 4000 个变量！我们可转而使用类的属性来存储每个气泡的这些值。

着手编写代码前，我们先来定义气泡。如本章前面的图 9-1 所示，气泡是彩色的实心圆，尺寸和位置各异。

所有这些都是气泡的属性。在面向对象编程中，我们在创建新类时根据描述对象的名词和形容词来确定属性清单。属性是存储在类中的变量。

除属性外，类还可包含方法。方法是与特定类相关联的函数，让类能够有所作为。请想一想，在本章的绘图应用中，我们希望气泡能做什么呢？我们要在用户单击或拖曳鼠标时创建气泡，还要在屏幕刷新时重绘气泡。诸如**创建**和**绘制**等动词可帮助我们确定气泡类需要包含哪些方法。

在一个类中包含所有必要的属性和方法后，就可描述任何要在屏幕上显示的气泡。这就是程序员使用面向对象编程时解决问题的方式：将较大的应用分成小块，通过自问程序包含什么来创建类，再自问每个类的对象需要做什么以及需要哪些信息，以确定类应包含哪些方法和属性。

下面来着手编写 Bubble 类。我们将在 BubblePanel 类中完成这项任务，因为我们只需在 BubblePanel 类中绘制气泡。请在 BubblePanel 类的右大括号前面定义 Bubble 类，如下所示：

```
import javax.swing.JPanel;
public class BubblePanel extends JPanel {
    private class Bubble {
    }
}
```

通常将内部助手类声明为私有的，以防其他程序直接访问它们。因此，我们将 Bubble 类声明为私有的，使其成为只能在 BubblePanel 中访问的内部类。这种方法被称为**封装**，意味着 BubblePanel 类向其他类隐藏了内部工作原理。封装是面向对象编程的核心原则之一，也是值得遵循的最佳实践，因为这意味着我们可以放心大胆地修改 Bubble 类的工作原理，而不会导致其他代码无法正常运行。由于 Bubble 类被声明为私有的，我们知道 BubblePanel 类外面的任何代码都不会依赖于它。对于包含大量类且由众多程序员协作开发的大型应用来说，这显得尤为重要。

用户在屏幕上绘画时，BubblePanel 将依赖于 Bubble 类来存储各个气泡的信息。气泡有表示其在屏幕上位置的(x, y)坐标，还有尺寸和颜色，我们可将这些属性作为 Bubble 类中的变量。对于气泡的 x 和 y 坐标以及尺寸，可存储为整数值：

```
import javax.swing.JPanel;
public class BubblePanel extends JPanel {
    private class Bubble {
        private int x;
        private int y;
        private int size;
    }
}
```

在这里，我们创建了两个变量（x 和 y）来存储气泡的坐标，还创建了变量 size。这些属性都被声明为私有的，因此只有 Bubble 类能够直接修改它们的值。我们将气泡的数据都封装在 Bubble 类中，并只使用这个类的方法来与气泡交互。

前面说过，在本章的应用中，每个气泡都有其颜色。java.awt 库包含一个 Color 类，能够使用 RGB（红–绿–蓝）颜色值来重现可在显示器上显示的任何颜色。本章后面编写控制气泡颜色的代码时，将更详细地介绍 RGB，因此现在直接在文件开头导入 java.awt.Color 类，并在 Bubble 类中添加一个名为 color 的属性，如下所示：

```
import java.awt.Color;
import javax.swing.JPanel;
public class BubblePanel extends JPanel {
    private class Bubble {
        private int x;
        private int y;
        private int size;
        private Color color;
    }
}
```

添加每个气泡都必须有的 4 个属性（x、y、size 和 color）后，就可着手实现给这些属性提

供值的行为或功能了。为此，我们将在 Bubble 类中添加方法。

9.3.2 设计 Bubble 类的方法

Bubble 类有两种行为——创建气泡以及在屏幕上绘制气泡，我们将把它们转换为方法。

创建对象的方法有一个特殊的名称：**构造函数**。构造函数通过给对象的属性赋值来设置对象。在构造函数中给属性赋值时，就是在**初始化**属性。对于 Bubble 类，我们要初始化每个气泡的属性 x、y、size 和 color。

1. 编写构造函数

构造函数以关键字 public 打头，接下来是类名和一对括号。要在创建对象时传递参数，可将它们放在括号内。我们要在创建每个气泡时都指定其 x、y 和 size 值，因此 Bubble 类的构造函数类似于代码清单 9-2。

代码清单 9-2 Bubble 类的构造函数

```
private class Bubble {
    private int x;
    private int y;
    private int size;
    private Color color;
 ❶ public Bubble(int newX, int newY, int newSize) {
     ❷ x = newX;
     ❸ y = newY;
     ❹ size = newSize;
    }
}
```

每个气泡的坐标和尺寸都将由用户决定，因此我们要在创建每个气泡时，都以整数值的方式将 *x* 和 *y* 坐标以及尺寸传递给构造函数 Bubble()。这些值是我们传递给方法 Bubble()的三个参数，分别名为 newX、newY 和 newSize❶。后面处理用户创建气泡的代码位于私有类 Bubble 外面，这意味着它们无法直接访问气泡的属性，因此无法给属性 x、y 和 size 赋值。为避开这种问题，我们通过 Bubble()构造函数给属性赋值，它接收来自用户的输入，将输入值赋给 newX、newY 和 newSize，再将用户输入值赋给气泡的属性（❷❸和❹）。

我们还要给每个气泡指定一种随机颜色。由于颜色是随机的，我们无需通过参数将其传递给构造函数，而可在构造函数中生成颜色。

为给每个气泡指定颜色，我们需要生成随机的 RGB 颜色值。在计算机显示器上，RGB 通过组合不同数量的红、绿、蓝光线来生成不同的颜色。在编程中，这三种颜色都用 0（无）到 255（最大可能值）的整数表示。要生成颜色，需要三个 0~255 的整数，它们放在括号内并用逗号分隔。例如，纯红色的 RGB 值为(255, 0, 0)，这意味着无绿色和蓝色，而红色为最大值；黄色的 RGB 值为(255, 255, 0)，这意味着无蓝色，但红色和绿色都为最大值。RGB 颜色值总共可表示超过 1600 万种颜色。

本书前面使用了 Math 类的方法 Math.random()来生成随机数，但那时只需要一个随机数，而在 BubbleDraw 应用中需要三个随机数（RGB 的每部分一个），因此这里介绍一种新的随机数生成方式。

类 java.util.Random 包含多个很有用的方法，其中的 nextInt()让我们能够立即生成不超过指定值的随机整数，而无需做额外的数学运算或圆整。为使用 Random 类，必须先在文件 BubblePanel.java 开头导入它：

```
import java.util.Random;
import java.awt.Color;
import javax.swing.JPanel;
```

由于 Random 是一个类，要访问其方法，必须创建一个类型为 Random 的对象或变量。请在 BubblePanel 类的开头添加如下代码行：

```
import java.util.Random;
import java.awt.Color;
import javax.swing.JPanel;
public class BubblePanel extends JPanel {
    Random rand = new Random();
    private class Bubble {
```

这行代码创建一个名为 rand 的随机数生成器，让我们能够快速生成随机整数或浮点数。我们将 rand 封装在 BubblePanel 类中，这样可在绘图窗口和内部类 Bubble 中使用它来生成随机数，但不能在 BubblePanel 外面使用它。

要生成随机颜色，可使用前面导入的 java.awt.Color 类的构造函数。这个构造函数接受三个 0~255 的整型参数，它们分别表示用于生成颜色的红色、绿色和蓝色值。

请在构造函数 Bubble()中添加如下代码：

```
        public Bubble(int newX, int newY, int newSize) {
            x = newX;
            y = newY;
            size = newSize;
            color = new Color( rand.nextInt(256),
                    rand.nextInt(256),
                    rand.nextInt(256) );
        }
```

我们使用关键字 new 和 Color 类的构造函数创建了一种新颜色，其中每个 RGB 颜色值都是随机生成的 0~255 的整数。每当你调用方法 nextInt()时，它都将生成一个随机整数，这个整数位于 0 和你传入的最大整数之间。在这里，我们要生成 0~255 的随机颜色值，因此传入 256，因为 nextInt()生成一个小于指定上限的随机整数。

2. 编写绘制气泡的方法

每个气泡都有(x, y)坐标、尺寸和随机颜色后，我们来添加在屏幕上绘制气泡的功能。为了在屏幕上绘制彩色图形，我们将导入 java.awt.Graphics 类。Graphics 类包含很多方法，如选择颜料颜色的 setColor()、绘制矩形的 drawRect()、绘制实心椭圆的 fillOval()等。我们将使用

fillOval()来绘制圆，因此请在文件 BubblePanel.java 开头添加如下 import 语句：

```
import java.awt.Graphics;
import java.util.Random;
import java.awt.Color;
import javax.swing.JPanel;
```

接下来，在内部类 Bubble 中（构造函数 Bubble()后面）添加方法 draw()：

```
    private class Bubble {
        private int x;
        private int y;
        private int size;
        private Color color;
        public Bubble(int newX, int newY, int newSize) {
            --snip--
        }
        public void draw(Graphics canvas) {
          ❶ canvas.setColor(color);
          ❷ canvas.fillOval(x - size/2, y - size/2, size, size);
        }
    }
```

方法 draw()接受一个参数——名为 canvas 的 Graphics（表示画布）。在方法 draw()中，我们对 canvas 调用方法 setColor()❶，将画笔颜色设置为存储在当前气泡的属性 color 中的颜色。在❷处，我们调用 fillOval()在屏幕上绘制一个实心圆。方法 fillOval()接受 4 个参数：要绘制的椭圆的定界框左上角的 x 和 y 坐标以及该外接矩形的宽度和高度。可将定界框视为这样一个不可见的矩形，即其左上角坐标为(x, y)，宽度和高度为指定的值；并将椭圆视为一个这样的气球，即它不断扩大，直到触及到定界框的 4 条边，如图 9-2 所示。通过将定界框的宽度和高度设置为相同的值，使其为正方形，绘制的将是一个圆而不是椭圆。

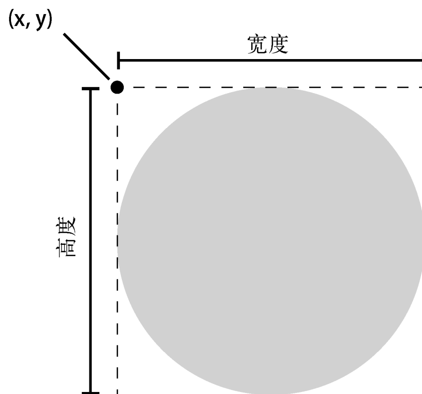


图 9-2　方法 fillOval()接受 4 个参数：左上角的 x 和 y 坐标以及椭圆的宽度和高度

由于要绘制的椭圆实际上是圆，因此宽度和高度相同，都是存储在 size 属性中的气泡尺寸

（单位为像素）。在❷处，我们要让气泡以用户单击的位置(x, y)为中心，因此相应地调整了气泡左上角的坐标，将它们分别设置为 x 和 y 减去气泡尺寸一半（size/2）的结果。

添加方法 draw()后，Bubble 类就编写好了。这个类能够记录气泡的位置、尺寸和颜色；我们还可使用它的方法来创建气泡以及在屏幕上绘制气泡。现在该在 BubblePanel 类中添加逻辑，在用户在屏幕上单击并拖曳时使用这些方法来创建并绘制气泡了。

9.4　将气泡存储在 ArrayList 中

用户在屏幕上单击并拖曳时，我们需要存储其创建的所有气泡。Java 库包含多个很有用的**数据结构**，它们都是用于存储一系列对象的类。

java.util.ArrayList 是一种**动态**数据结构，这意味着它不仅能够存储一系列对象，还能根据程序的需要增大或缩小。在 BubbleDraw 应用中，我们无法预测用户会绘制多少个气泡，因此为存储用户创建的所有气泡，像 ArrayList 这样的动态数据结构是理想的选择。ArrayList 是一种灵活的存储方式，适合用于预先不知道要存储多少对象的情形。在 Java 中，普通数组的长度是固定的，但使用 ArrayList 时，可随用户不断单击鼠标将气泡添加到其中。

首先，在文件 BubblePanel.java 开头导入 java.util.ArrayList，以便能够使用这种数据类型：

```
import java.util.ArrayList;
import java.awt.Graphics;
import java.util.Random;
import java.awt.Color;
import javax.swing.JPanel;
```

接下来，需要声明一个 ArrayList 变量，用于存储 Bubble 对象。为此，在 BubblePanel 类中添加如下声明：

```
public class BubblePanel extends JPanel {
    Random rand = new Random();
    ArrayList<Bubble> bubbleList;
```

声明 ArrayList 变量时，可在尖括号<和>之间使用类型说明符，告诉 Java 该变量将用于存储哪种类型的对象。ArrayList 可存储任何类型的对象，但这里声明的 ArrayList 变量 bubbleList 只能存储 Bubble 对象。

接下来，再声明一个变量——size，用于存储气泡的默认尺寸：

```
public class BubblePanel extends JPanel {
    Random rand = new Random();
    ArrayList<Bubble> bubbleList;
    int size = 25;
```

这个 size 变量用于指定气泡的默认尺寸——单位为像素的直径。我指定的是 25 像素，但你可根据喜好选择更大或更小的值。

9.4.1 给 BubblePanel 类添加构造函数

将 bubbleList 声明为存储 Bubble 对象的 ArrayList 后，我们给 BubblePanel 类添加一个构造函数，以初始化 bubbleList 并设置绘图窗口的背景色。

与内部类 Bubble 的构造函数一样，BubblePanel 类的构造函数声明也由 public、类名和一对括号组成：

```
public class BubblePanel extends JPanel {
    Random rand = new Random();
    ArrayList<Bubble> bubbleList;
    int size = 25;
    public BubblePanel() {
    }
```

注意，我们在这个构造函数末尾添加了左右括号，因为构造函数也是方法。在这个构造函数中，我们初始化 bubbleList，为存储 Bubble 对象做准备；还将绘图窗口的背景色设置为黑色：

```
    public BubblePanel() {
     ❶ bubbleList = new ArrayList<Bubble>();
     ❷ setBackground(Color.BLACK);
    }
```

注意 虽然我推荐将这个应用的背景色设置为黑色，但为让你看清楚，本书显示的图像的背景都为白色。

与声明 bubbleList 时一样，创建 ArrayList 时，可在尖括号内指定要存储的对象类型：<Bubble>❶。

在❷处，我们之所以能够直接使用方法 setBackground()，是因为 BubblePanel 扩展了 JPanel，而我们知道 JPanel 有背景色。我们将背景色设置为颜色常量 Color.BLACK，这个常量位于前面导入的 Color 类中，其 RGB 值为(0, 0, 0)。

现在，当应用 BubbleDraw 启动并创建 BubblePanel 时，将有一个空的气泡列表，而屏幕的背景色为黑色。

请保存所做的修改。在下一节，我们将在窗口中填充气泡。

9.4.2 添加在屏幕上绘图的方法

接下来，我们需要添加一个方法，它将 bubbleList 中的所有气泡都绘制到屏幕上。

javax.swing 工具包中的所有 GUI 组件（包括表示画布的 BubblePanel 扩展的 JPanel）都有方法 paintComponent()，它在屏幕上绘制组件。我们将修改（**重写**）默认的 paintComponent()方法，让 BubblePanel 绘制 bubbleList 中的所有气泡。

首先，需要声明方法 paintComponent()。由于我们要重写 BubblePanel 的父类 JPanel 的方法

paintComponent(),因此必须使用相同的**方法签名**(即第1行必须相同)。这意味着 paintComponent() 的声明必须与原来完全相同,即将其声明为 public void 的,且接受一个类型为 Graphics 的参数:

```
public class BubblePanel extends JPanel {
    --snip--
    }
    public void paintComponent(Graphics canvas) {
    }
```

我们将方法 paintComponent()放在了构造函数 BubblePanel()的后面。请注意,paint Component()接受一个 Graphics 参数;与为 Bubble 类编写的方法 draw()一样,我们将这个参数命名为 canvas。任何在屏幕上绘画的对象都可使用 Graphics 来给计算机屏幕上的像素着色。

在这个方法中,我们要让父类 JPanel 清屏并执行绘图前的其他准备工作。为此,我们调用这个父类的方法 paintComponent():

```
    public void paintComponent(Graphics canvas) {
        super.paintComponent(canvas);
    }
```

关键字 super 让 Java 调用 JPanel 类的方法 paintComponent(),这意味着原始 paintComponent() 方法的所有代码都将加入到新的 paintComponent()方法中。这是一项很有用的面向对象编程功能:由于我们扩展了 JPanel 类来创建新类 BubblePanel,因此可利用 JPanel 的所有功能,如在应用启动时清除窗口中的像素以及为绘制彩色图形准备好 Graphics。在这里,我们说 BubblePanel **继承**了 JPanel 的这些功能。

准备好画布后,该遍历气泡列表并将它们绘制到画布上了。为此,我们将以你以前没见过的方式使用 for 循环。

在本书中,第 6 章首次使用了 for 循环来遍历字符串中的字符,以便对消息进行加密。这次我们将使用 for 循环的简化版——专门为遍历对象列表或集合而设计的 for each 语句。

我们先来看看代码,再详细地介绍 for each 语句的各个部分:

```
    public void paintComponent(Graphics canvas) {
        super.paintComponent(canvas);
     ❶ for(Bubble b : bubbleList) {
         ❷ b.draw(canvas);
        }
    }
```

对于❶处的 for each 语句,你可将其解读为“对于 bubbleList 中的每个 Bubble 对象 b”。你之所以知道这是一条 for each 语句而不是普通 for 循环,原因有两个。首先,这条语句中间有冒号;其次,它不包含第 6 章介绍的 for 循环的三个部分:**初始化**、**条件**和**更新**。在 Java 中,这两种循环都使用关键字 for 和括号,但 for each 语句是专门为遍历对象集合(如数组、ArrayLists 等)而设计的。

对于 ArrayList bubbleList 中的每个 Bubble 对象 b,这个循环都将调用 b.draw(canvas)将其

绘制到画布上❷。这个循环第一次执行时，b 指向的是 bubbleList 中的第一个 Bubble 对象，每次重复该循环时，b 都将指向这个列表中的下一个气泡。b.draw(canvas)让气泡将自己绘制到画布上。

这个简短的 for each 循环将把 bubbleList 中的每个气泡都绘制到屏幕上。唯一的问题是，我们没有可供用来测试的气泡。下面来生成一些随机气泡，看看这个应用的运行情况，再接着添加让用户能够使用鼠标创建气泡的代码。

9.4.3 测试 BubblePanel 类

编写支持鼠标交互的代码前，我们先来测试一下已编写的代码。为此，我们将编写一个绘制 100 个气泡的测试方法，这些气泡的尺寸是随机的，它们分布在应用窗口的各个地方。这让我们能够在确定这个应用管用后，再编写最终的代码。这样，我们既能尽早找出代码中的错误，又能预先知道最终的应用有多酷。

将这个新方法命名为 testBubbles()，并将其放在方法 paintComponent()和 private class Bubble 之间，如代码清单 9-3 所示。

代码清单 9-3 创建方法 testBubbles()

```
    public void paintComponent(Graphics canvas) {
        --snip--
        }
    }
❶ public void testBubbles() {
    ❷ for(int n = 0; n < 100; n++) {
        ❸ int x = rand.nextInt(600);
        ❹ int y = rand.nextInt(400);
        ❺ int size = rand.nextInt(50);
        ❻ bubbleList.add( new Bubble(x, y, size) );
        }
    ❼ repaint();
    }
    private class Bubble {
```

在❶处，我们将 testBubbles()声明为公有的且返回类型为 void，这意味着它不会返回任何信息。接下来，我们使用一个普通 for 循环从 0 迭代到 99——总共迭代 100 次❷。这意味着我们将创建 100 个气泡，而对于每个气泡，我们都需要(x, y)坐标以及以像素为单位的尺寸（用于指定气泡的宽度和高度）。

在本章开头，我们将应用 BubbleDraw 的窗口设置成了 600 像素宽、400 像素高，因此对于气泡的位置，x 值必须在范围 0~600 内，而 y 值必须在范围 0~400 内。在这个 for 循环中，我们使用随机数生成器 rand 来生成一个 0~600 的随机整数，并将其存储在表示气泡中心 x 坐标的变量 x 中❸。然后，生成一个 0~400 的 y 坐标值，并将其存储到变量 y 中❹。接下来，为指定以像素为单位的气泡尺寸，我们生成一个 0~50 的随机整数❺。这个循环中的最后一步是，使用刚生

成的随机值 x、y 和 size 创建一个 Bubble 对象，并将其添加到 ArrayList bubbleList 中❻。

最后，调用方法 repaint()❼。通常，绘制新的计算机图形前，必须清屏——绘制空空如也的黑色背景，但这里 repaint()替我们完成了这项工作。请注意，在方法 paintComponent()中也无需清屏，而只是绘制 bubbleList 中的所有气泡。方法 repaint()负责重绘背景并调用 paintComponent()，因此每当我们要刷新或重绘屏幕时都调用它。

再完成一个步骤我们就可测试这个应用了：在构造函数 BubblePanel()中调用方法 testBubbles()。为此，请在 BubblePanel 类的构造函数中添加下面这样的代码：

```
public BubblePanel() {
    bubbleList = new ArrayList<Bubble>();
    setBackground(Color.BLACK);
    testBubbles();
}
```

将文件 BubblePanel.java 存盘，再切换到选项卡 BubbleDraw.java。将这个文件也存盘，再单击 Run 按钮。首次编译并运行应用时，必须在选项卡 BubbleDraw.java 中进行，因为运行程序的方法 main()包含在 BubbleDraw 中。你将看到一个窗口，其中充斥着位置随机且五颜六色的气泡，如图 9-3 所示。

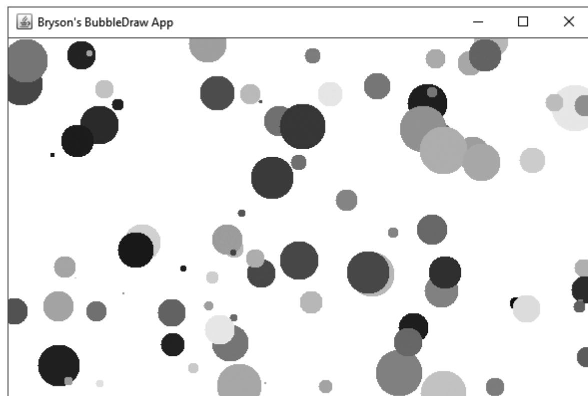


图 9-3　每次运行这个版本的 BubbleDraw 应用时，方法 testBubbles()都将在屏幕上绘制 100 个气泡

如果你喜欢随处飘散的气泡，花点时间尝试调整方法 testBubbles()中使用的数字。看看你能否绘制 200、500 乃至 1000（而不是 100）个气泡。让气泡更大些，为此可生成更大的尺寸随机数，或给当前生成的尺寸随机数加上一个值。调整 x 和 y 值，看看你能够让所有气泡都在屏幕内，而不是让有些气泡被屏幕边缘裁剪掉一部分。

想怎么修改方法 testBubbles()就怎么修改，这可是尝试新鲜事物并立即查看修改效果的绝佳机会。下一节添加鼠标交互功能时，我们将把调用方法 testBubbles()的代码注释掉，因此你可随便尝试修改方法 testBubbles()，而无需担心把程序其他部分搞得一团糟。

9.5 处理鼠标事件

应用 BubbleDraw 的目标是让用户能够使用鼠标来绘制气泡。我们在前一节看到，绘制气泡的代码是管用的，现在只需添加支持鼠标交互的代码。

我们将使用事件监听器来让这个应用能够处理鼠标单击、鼠标移动和鼠标滑轮滚动。在第 3 章和第 7 章的 GUI 应用中，我们使用了匿名内部类来添加事件监听器，以处理按钮单击以及滑条和文本框变化，但如果在这个应用中也使用匿名内部类，每种事件都将需要三个监听器，这将难以跟踪。另外，对于这些事件中的两个——单击鼠标和拖曳鼠标，我们想做相同的处理：让应用在用户单击鼠标和拖曳鼠标时都添加气泡。相比于为一个事件编写监听器，再将其代码复制并粘贴到另一个事件的监听器中，将同一个代码块关联到这两个事件要容易得多。因此，为让这个应用更易于管理，我们将创建单个具名事件监听器，它能够响应前述全部三种事件。

9.5.1 创建一个可重用的事件监听器

要创建事件监听器，必须在文件 BubblePanel.java 开头再导入一个库——java.awt.event.*：

```
import java.awt.event.*;
import java.util.ArrayList;
import java.awt.Graphics;
import java.util.Random;
import java.awt.Color;
import javax.swing.JPanel;
```

我们以前编写的 import 语句都只导入一个类，但这条语句导入 java.awt.event 库中的所有类。星号（*）是一个通配符，表示要导入 java.awt.event 顶层的所有类，其中包括所有的鼠标事件和监听器。原本可在用到每个类时分别导入，但通过导入 java.awt.event.*，程序编写起来将容易得多，因为这样可专注于编写代码，而无需在每次用到新的事件类时都切换到文件开头。

下面来编写私有的具名内部类，以监听鼠标事件。我们将它命名为 BubbleListener，因为它将处理 BubblePanel 类中所有与气泡相关的事件。请将这个类放在方法 testBubbles()和私有类 Bubble 之间。Java 程序员通常将监听器类和其他辅助类一起放在类文件末尾附近，这种约定旨在让我们能够在调试或修改文件时快速找到监听器代码。

```
        repaint();
    }
    private class BubbleListener extends MouseAdapter {
    }
    private class Bubble {
```

BubbleListener 类扩展了处理鼠标事件的 MouseAdapter 类。使用关键字 extends 基于 JFrame 创建新类时，新类将继承父类 JFrame 的所有属性和函数，同样，BubbleListener 也将继承 MouseAdapter 类的所有鼠标事件监听器功能。这个适配器类能够处理单击事件（MouseListener）、鼠

标移动事件（MouseMotionListener）以及使用鼠标滑轮或触控板执行的滚动事件（MouseWheelListener）。

我们将在代码中分步添加这些事件处理程序，并在每次添加后都进行测试，看看应用 BubbleDraw 增加功能后的情况。

9.5.2　处理单击和拖曳

处理鼠标事件时，需要分两步进行。首先，在 BubbleListener 中添加处理事件的代码，如 mousePressed()。然后，在构造函数 BubblePanel()中添加监听器，让 BubblePanel 监听指定类型的事件，并在该事件发生时调用 BubbleListener 来处理。下面来处理第一种情形——用户单击鼠标以绘制气泡。

1. 监听鼠标按钮事件

可监听的鼠标按钮事件有三个。

- mousePressed()：在用户按下鼠标按钮时发生。
- mouseReleased()：在用户松开鼠标按钮时发生。
- mouseClicked()：在用户快速按下并松开鼠标时发生。

对于应用 BubbleDraw，我们将使用处理程序 mousePressed()，让应用在用户按下鼠标按钮时就绘制一个气泡。事件处理程序的 mousePressed()签名类似于下面这样：

```
private class BubbleListener extends MouseAdapter {
    public void mousePressed(MouseEvent e) {
    }
}
```

注意　编写事件处理程序时，拼写和大小写显得尤其重要，因为它们是具有特定大小写的内置名称。请确保你的方法 mousePressed()与这里列出的完全相同。

事件处理程序 mousePressed()位于 BubbleListener 类中，必须将其声明为公有的且返回类型为 void，以便与 MouseAdapter 类的方法 mousePressed()一致。另外，它接受一个类型为 MouseEvent 的参数。所有鼠标事件都获取事件发生时鼠标在屏幕上的位置信息。鼠标事件的 x 和 y 坐标存储在一个 MouseEvent 对象中，可使用方法 getX()和 getY()来获取这些坐标。

下面的代码在 bubbleList 中添加一个气泡（其位置为用户单击鼠标的位置），再重绘屏幕让这个气泡出现。

```
private class BubbleListener extends MouseAdapter {
    public void mousePressed(MouseEvent e) {
     ❶ bubbleList.add(new Bubble(e.getX(), e.getY(), size));
     ❷ repaint();
    }
}
```

在❶处，我们创建了一个 Bubble 对象，其(*x*, *y*)位置为用户单击鼠标的位置——e.getX()和 e.getY()。MouseEvent 类有多个用于处理鼠标事件的属性和方法，如确定用户按下的是哪个鼠标按钮以及鼠标在屏幕上的位置。刚才说过，方法 getX()和 getY()提供鼠标事件（如单击或拖曳）的 *x* 和 *y* 坐标。如果你查看 Bubble 类，将发现我们为它编写的构造函数接受三个参数——int newX、int newY 和 int newSize，因此我们必须将 e.getX()、e.getY()和 size 传入构造函数以新建一个 Bubble 对象。创建气泡后，我们将使用 bubbleList.add()将其添加到存储气泡的 ArrayList 中。

在❷处，我们调用方法 repaint()来刷新屏幕，并将更新后的 bubbleList 绘制到画布上。

这就编写好了事件处理程序 mousePressed()，但我们还需做一件事情，以便让应用监听 mousePressed()事件，并将其交给 BubbleListener 类进行处理。我们必须让 BubblePanel 类将 BubbleListener 作为鼠标事件监听器。

为此，在构造函数 BubblePanel()中做两项修改。首先，将调用 testBubbles()的代码行注释掉：在它前面加上两个斜杠。其次，调用方法 addMouseListener()指定使用 BubbleListener 来处理鼠标事件。更新后的构造函数如下所示：

```
public BubblePanel() {
    bubbleList = new ArrayList<Bubble>();
    setBackground(Color.BLACK);
 ❶ // testBubbles();
 ❷ addMouseListener( new BubbleListener() );
}
```

通过将调用方法 testBubbles()的代码注释掉❶，可禁止这个方法执行，以便能够使用鼠标来测试交互式应用 BubbleDraw，同时将这个方法保留下来，以防以后还要绘制随机的测试气泡。❷处新增的代码行让 BubblePanel 监听鼠标事件，并在这些事件发生时交给 BubbleListener 类去处理。

完成这些修改后，就可运行应用 BubbleDraw 并使用鼠标在单击的地方添加气泡，如图 9-4 所示。

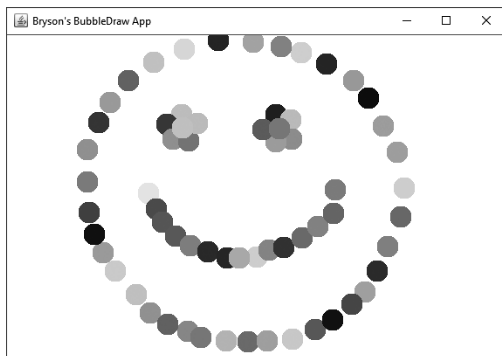


图 9-4 添加处理程序 mousePressed()，并将 BubbleListener 指定为鼠标事件监听器后，就可通过单击在屏幕的任何地方绘制气泡

很好！你可反复单击，在屏幕上绘制形状和图案，但如果能拖曳鼠标来绘制气泡将更容易。下面就来添加这种功能。

2. 监听鼠标移动事件

鼠标移动事件和鼠标按钮按下事件属于不同的类型，但也可在扩展 MouseAdapter 的类（如 BubbleListener）中进行处理。鼠标移动事件有两种：mouseMoved()和 mouseDragged()。

mouseMoved()事件在鼠标在绘图窗口中移动时发生，而 mouseDragged()事件在用户按下鼠标按钮并移动鼠标时发生。由于我们只想在用户单击并拖曳时绘制气泡，因此将首先在 BubbleListener 类中实现事件处理程序 mouseDragged()，再在 BubblePanel 的构造函数添加一个鼠标移动监听器来激活监听器 BubbleListener。

请在 BubbleListener 类中添加事件处理程序 mouseDragged()，如下所示：

```
private class BubbleListener extends MouseAdapter {
    public void mousePressed(MouseEvent e) {
        --snip--
    }
    public void mouseDragged(MouseEvent e) {
        bubbleList.add(new Bubble(e.getX(), e.getY(), size));
        repaint();
    }
}
```

你应该注意到了，这些代码与事件处理程序 mousePressed()几乎完全相同，只是名称为 mouseDragged()。这是因为它们都处理 MouseEvent。方法 mousePressed()在用户按下鼠标按钮时处理事件，而 mouseDragged()在用户拖曳鼠标时被调用。在这个应用中，响应这两种事件的方式相同：在 bubbleList 中添加气泡，再调用函数 repaint()。

接下来，在构造函数 BubblePanel()中使用下面的语句将 BubbleListener 指定为鼠标移动事件监听器：

```
public BubblePanel() {
    bubbleList = new ArrayList<Bubble>();
    setBackground(Color.BLACK);
    // testBubbles();
    addMouseListener( new BubbleListener() );
    addMouseMotionListener( new BubbleListener() );
}
```

同样，我们来运行应用以测试这个新增的事件监听器。将代码存盘，单击 Run 按钮，再在出现的应用窗口中四处单击并拖曳，如图 9-5 所示。



图 9-5 现在可单击并拖曳以绘制一连串气泡!

厉害吧?仅通过添加两个事件处理程序(它们分别处理鼠标按钮按下事件和鼠标移动事件),就创建了一个多姿多彩的交互式绘图应用。

9.5.3 处理鼠标滑轮事件

至此,我们添加了两个短小精悍的事件处理程序,它们分别处理按下鼠标按钮和单击并拖曳鼠标事件,但 MouseAdapter 还有一个事件类:MouseWheelEvent。根据你的系统,它可能配置了带滑轮的鼠标,也可能配置的是触控板,可用来在文档和网页中上下滚动。你可能使用的是笔记本电脑,配置了让你能够使用手势来滚动的触控板;例如,在 MacBook 上为两个手指轻扫手势,而在大多数 Windows 笔记本电脑上为沿触控板右边缘轻扫。

Java 将这两种滚动事件都视为 MouseWheelEvent。我们可在 BubbleListener 类中添加一个 mouseWheelMoved()处理程序,以处理鼠标滑轮事件和触控板滚动手势事件,如下所示:

```
private class BubbleListener extends MouseAdapter {
    public void mousePressed(MouseEvent e) {
        --snip--
    }
    public void mouseDragged(MouseEvent e) {
        --snip--
    }
❶  public void mouseWheelMoved(MouseWheelEvent e) {
      ❷ size += e.getUnitsToScroll();
    }
}
```

在❶处,方法 mouseWheelMoved()接受一个 MouseWheelEvent 参数。在❷处,我们使用 e.getUnitsToScroll()从 MouseWheelEvent 参数 e 那里获取鼠标滑轮滚动了多少个单位,并将 size 增加相应的量。getUnitsToScroll()返回的数字可能为正,也可能为负,这取决于你使用的操作系统以及滑轮是向下还是向上滚动的。

不同操作系统在处理滚动方面的差异

Windows、macOS 和 Linux 在处理滚动方面存在差异。在 macOS 系统中，向上滚动或轻扫将返回正值，因此用户向上滚动或轻扫时，刚才编写的代码将增大气泡，而用户向下滚动时将缩小气泡。在 Windows 和大多数 Linux 系统中，使用鼠标滑轮向上滚动将向下移动网页或文档，而 getUnitsToScroll()将返回负值。这意味着向上滚动将缩小气泡，而向下滚动将增大气泡。

要让这个应用在全部三种操作系统中的行为都相同，可像下面这样修改代码：

```
public void mouseWheelMoved(MouseWheelEvent e) {
    if(System.getProperty("os.name").startsWith("Mac"))
        size += e.getUnitsToScroll();
    else
        size -= e.getUnitsToScroll();
}
```

其中的 if 语句使用 System.getProperty ("os.name")获取操作系统的名称，并检查这个名称是否以字符串"Mac"打头。如果是，就将 size 加上滚动的单位数；否则将 size 减去滚动的单位数，从而在 Windows 和 Linux 系统中也在向上滚动时增大气泡，的向下滚动时缩小气泡。

最后，在构造函数 BubblePanel()中，在添加前面两个监听器的代码后面，添加鼠标滑轮监听器：

```
public BubblePanel() {
    bubbleList = new ArrayList<Bubble>();
    setBackground(Color.BLACK);
    // testBubbles();
    addMouseListener( new BubbleListener() );
    addMouseMotionListener( new BubbleListener() );
    addMouseWheelListener( new BubbleListener() );
}
```

现在保存并运行应用，你将能够通过滚动鼠标滑轮或在触控板上轻扫来调整气泡的尺寸，如图 9-6 所示。

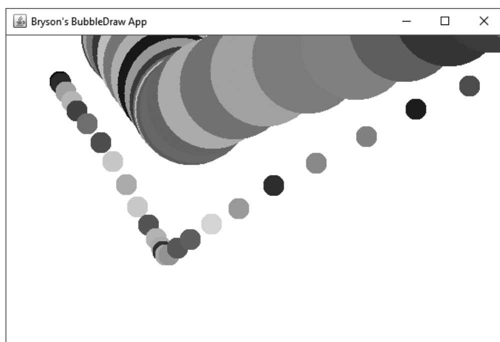


图 9-6 使用鼠标滑轮或触控板来调整气泡的尺寸

第一个版本的 BubbleDraw 应用创建好了！你可通过单击和拖曳在屏幕上绘制颜色各不相同的气泡，通过上下滚动来调整气泡的尺寸，将窗口最大化在全屏幕时下绘图等。

9.6 小结

你采用面向对象的编程方法，创建了一个 Bubble 类来定义气泡的属性和行为，从而开发了一个多姿多彩的交互式绘图应用。

下面是你在本章学到的一些新技能：

- 创建包含多个类和多个 Java 源代码文件的应用；
- 使用面向对象的编程方法将问题分解为较小的片段；
- 从头开始创建包含属性、方法和构造函数的 Bubble 类；
- 使用 java.util.Random 和 java.awt.Color 生成随机颜色；
- 使用 Random.nextInt(range)生成指定范围内的随机数；
- 使用 java.awt.Graphics 类绘制图形；
- 使用方法 paintComponent()指定应用要绘制的内容，使用 repaint()来清屏和重绘；
- 为鼠标事件（包括单击事件、移动事件和鼠标滑轮事件）编写事件处理程序；
- 使用 java.util.ArrayList 类存储动态对象（如应用 BubbleDraw 中的气泡）列表；
- 使用自定义监听器类 BubbleListener 在应用中添加多个事件监听器。

9.7 编程练习

为复习并使用学到的知识以及获得更多的编程技能，请尝试完成这里的编程练习。如果遇到困难，可从本书配套网站（https://www.nostarch.com/learnjava/）下载示例解决方案。

9.7.1 编程练习 1：避免气泡太小

如果你向下滚动鼠标滑轮的程度足够大，气泡将完全消失，因为 size 值将降低到小于零，

导致 Java 无法在屏幕上绘制相应的圆。实际上，`size` 值小于等于 2 后，气泡的尺寸将只有 1 像素，因此为让气泡是圆形的，必须确保气泡尺寸至少为 3 像素 × 3 像素。

在文件 BubblePanel.java 中，在 `BubbleListener` 类的方法 `mouseWheelMoved()`中添加一点逻辑，检查变量 `size` 是否小于 3，如果是，就将 `size` 设置为 3，确保气泡在屏幕上可见。请将这条 `if` 语句放在 `mouseWheelMoved()`方法末尾，从而在 `size` 发生变化之后（而不是之前）检查它。

修改后运行应用，你将能够滚动鼠标滑轮将气泡缩小到只有两像素大，但无法让它消失。干得漂亮！

9.7.2　编程练习 2：PixelDraw

只需在创建 `Bubble` 对象的构造函数中执行一些数学运算，就可实现很酷的像素化效果，如图 9-7 所示。

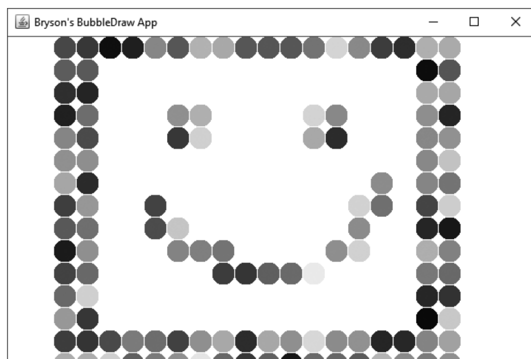


图 9-7　只需稍微调整新建气泡的位置，就可实现很酷的像素绘画效果

我们复制项目文件夹 BubbleDraw，再在复制的项目中进行修改，以免破坏原来的应用。为此，双击选项卡 BubblePanel.java 恢复到默认视图，在 Package Explorer 中选择项目文件夹 BubbleDraw，并按 Ctrl-C 或⌘-C 复制它，再按 CTRL-V 或⌘-V 将文件夹粘贴到工作区。将文件夹命名为 PixelDraw，如图 9-8 所示。

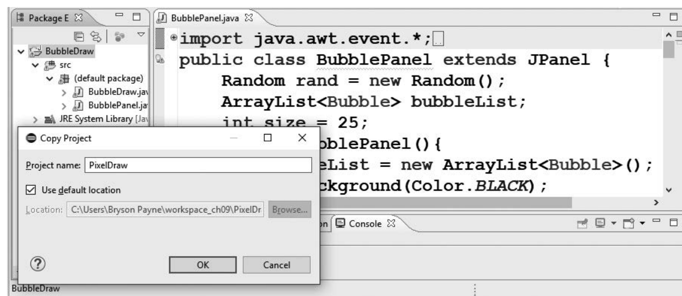


图 9-8　可在 Package Explorer 中复制并粘贴项目文件夹来创建新副本

将私有类 Bubble 的构造函数 Bubble()的代码修改成下面这样：

```
public Bubble(int newX, int newY, int newSize) {
    x = (newX / newSize) * newSize + newSize/2;
    y = (newY / newSize) * newSize + newSize/2;
    size = newSize;
    color = new Color( rand.nextInt(256),
            rand.nextInt(256),
            rand.nextInt(256) );
}
```

这些新代码只是修改了气泡的(*x*, *y*)位置，这是 Java 处理整数除法的方式导致的。当你将两个整数相除时，Java 只保留结果的整数部分，因此除以 newSize 再乘以 newSize 将导致气泡与网格对齐，如图 9-7 所示。例如，如果 newSize 为 10，而 x 为 25，则将 x 除以 newSize（25 / 10）的结果为 2，再乘以 10 后结果为 20，导致气泡与 10×10 的格子对齐。我们给每个坐标都加上 newSize/2，让气泡刚好位于格子内。别忘了，滚动鼠标滑轮依然会改变气泡的尺寸，因此你可绘制出稠密的块状图像，也可绘制气泡互不重叠的图形。

如果你要让这种像素化效果更具 Minecraft 风格，可让气泡为方形的。为此，可修改方法 draw()，使其绘制实心矩形而不是实心椭圆——将调用 fillOval()的代码注释掉，并添加调用 fillRect()的代码：

```
public void draw(Graphics canvas) {
    canvas.setColor(color);
    // canvas.fillOval(x - size/2, y - size/2, size, size);
    canvas.fillRect(x - size/2, y - size/2, size, size);
}
```

现在你可以以像素化风格绘画了，如图 9-9 所示。

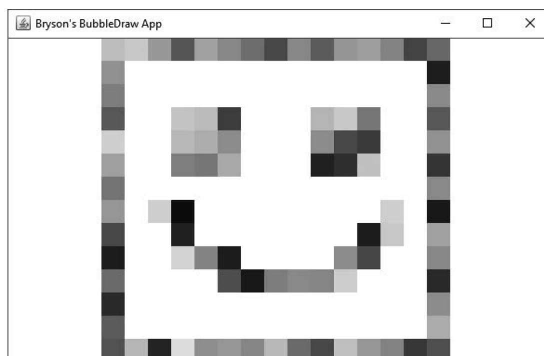
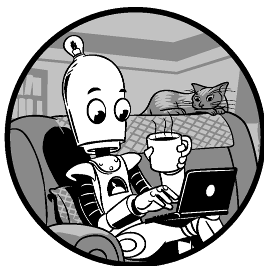


图 9-9　绘制矩形而不是椭圆来实现像素化绘画效果

9

第 10 章

添加动画和碰撞检测



本章将在应用 BubbleDraw 中添加基于定时器的动画，让气泡漂浮并能够回弹。我们还将给这个应用加上对用户友好的 GUI。改进后的应用名为 BubbleDrawGUI，我们将在其中添加一个 `JPanel`，它包含如图 10-1 所示的 GUI 组件，让用户能够启动和停止动画、修改动画速度以及清屏。

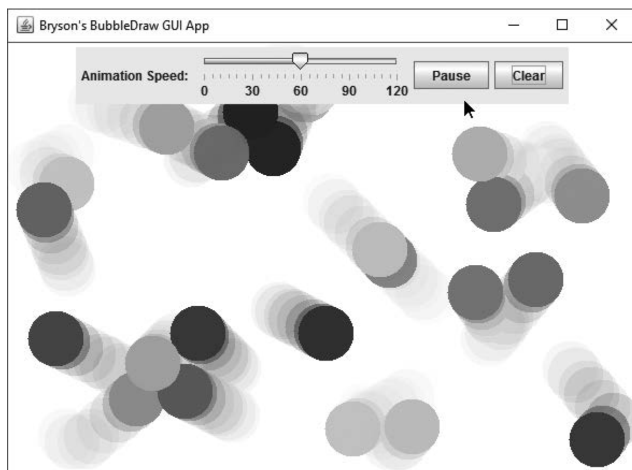


图 10-1　在改进版应用 BubbleDrawGUI 中，气泡是漂浮而半透明的且能够回弹；它还有让用户能够控制动画的 GUI

这个版本的应用交互性更强，对用户更友好，用户可通过单击和拖曳鼠标来绘制漂浮且能够回弹的气泡。

10.1　通过复制项目 BubbleDraw 来创建 BubbleDrawGUI

我们将在第 9 章的项目 BubbleDraw 的基础上创建这个新的 GUI 应用，因此这里不从头开始

新建 Java 项目，而是复制、粘贴并重命名项目 BubbleDraw。在你需要扩展既有程序来创建新版本，同时要保留原来的程序时，这种方法很有用。

在 Eclipse 中，右击 Package Explorer 中的项目文件夹 BubbleDraw 并选择 Copy，再在 Package Explorer 中右击并选择 Paste。这将打开 Copy Project 对话框，让你能够给复制的项目指定名称。输入 `BubbleDrawGUI` 并单击 OK 按钮，Eclipse 将在 Package Explorer 中创建项目 BubbleDraw 的副本，并将其命名为 BubbleDrawGUI。

10.1.1 重命名主类及其 Java 文件

现在重命名文件 BubbleDraw.java。这个文件包含运行应用的方法 `public static void main()`，将其重命名有助于区分新应用和旧版本。在项目文件夹 BubbleDrawGUI 中，右击 BubbleDraw.java 并选择 Refactor▸Rename。

重构意味着调整代码的结构，但不改变其功能。想出更佳、更高效的方法后，程序员通常对代码进行重构。在出现的 Rename Compilation Unit 对话框中，输入新名称 `BubbleDrawGUI`，再单击 Finish 按钮。可能还会出现一个警告窗口，指出这个类包含方法 `main()`。你可忽略这种警告，再次单击 Finish 按钮。重构过程将把类和 Java 文件都重命名为 BubbleDrawGUI。现在暂时保留 `BubblePanel` 类不变。

最后，修改 `JFrame` 窗口的标题，使其与新的 GUI 版本相称。请打开文件 BubbleDrawGUI.java，找到创建 `JFrame` 的代码行，并将其中的字符串改为 *`Your Name`*`'s BubbleDraw GUI App`，如下所示：

```
import javax.swing.JFrame;
public class BubbleDrawGUI extends JFrame {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Your Name's BubbleDraw GUI App");
```

保存文件并运行它，你将在窗口顶部的标题栏中看到新标题，如图 10-2 所示。

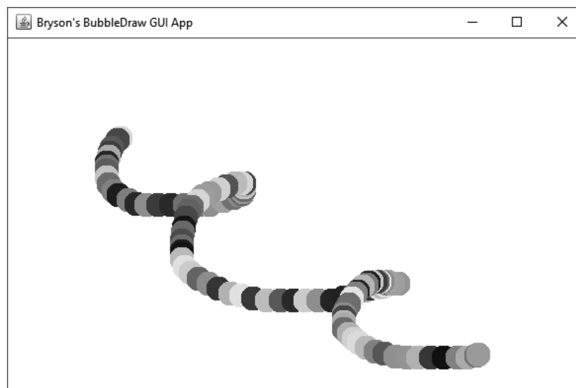


图 10-2 文件 BubbleDrawGUI.java 打开一个包含标题“*Your Name*'s BubbleDraw GUI App”的窗口

注意　首次运行包含多个文件的应用，如 BubbleDraw 和 BubbleDrawGUI 时，必须运行包含方法 main()的文件。运行主文件将创建配置，让你以后只需单击 Run 按钮就能运行程序。BubblePanel 类没有方法 main()，因此我们必须运行 BubbleDrawGUI.java 或右击项目文件夹 BubbleDrawGUI 并选择 Run As▶Java Application。

下面再做一项修改，让气泡看起来更逼真。

10.1.2　指定透明度

真实的气泡通常是半透明的。想想吃泡泡糖时吹气泡的情形吧：气泡足够大后，就能透过很薄的气泡表面看到气泡后面的景象。在应用 BubbleDrawGUI 中，可指定气泡的透明度，让它们看起来更逼真。

除第 9 章介绍的 RGB 颜色组分外，Java 还可在 java.awt.Color 类中存储第 4 个组分。这个组分名为 alpha，表示颜色绘制在其他物体前面时有多透明。与 RGB 颜色值一样，alpha 组分的取值范围也是 0~255。alpha 值为 0 时，颜色是看不见的；alpha 值为 128 时，颜色将像水彩画颜料一样是半透明的；alpha 值为 255 时，将把后面的物体完全遮住。

Color 类的构造函数可将 alpha 值作为第 4 个参数（紧跟在 RGB 值后面），因此我们只需修改文件 BubblePanel.java 中的一行代码就能指定透明度。请打开项目 BubbleDrawGUI 中文件夹 src 下的文件 BubblePanel.java，并滚动到这个文件末尾（定义 Bubble 类的地方）：

```
private class Bubble {
        private int x;
        --snip--
            color = new Color(rand.nextInt(256),
                    rand.nextInt(256),
                    rand.nextInt(256),
                    rand.nextInt(256) );
        }
```

在这里，我们修改了给变量 color 赋值的构造函数调用代码——添加了第 4 个随机值（其取值范围为 0~255）。我们在第三个 rand.nextInt(256)后面加上一个逗号，并在这个逗号和构造函数 Color()的右括号之间添加了第 4 个 rand.nextInt(256)。请务必仔细检查逗号和括号，确保它们与这里显示的一致，否则应用将无法正确运行。

保存这个文件并运行应用，在屏幕上单击绘制一些稍微有点重叠的气泡，如图 10-3 所示。

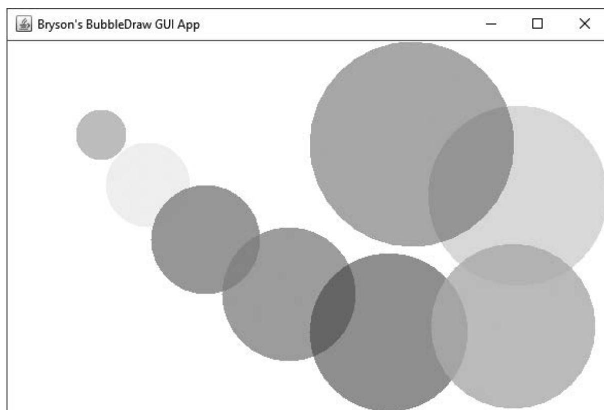


图 10-3 在每个气泡的颜色中加上 alpha 组分，让气泡看起来是透明的

你将发现，现在气泡不仅颜色是随机的，透明度也各不相同。有些气泡是不透明的，将后面的气泡完全遮住了，而有些透明，几乎看不到。现在的气泡更像气泡了！下面让气泡飘动起来，使其更逼真。

10.2 添加动画让气泡往上飘

动画是通过在屏幕上显示一系列图像营造出来的假象。你可能在备忘录中创建过翻书式动画：每个画面都与前一个画面稍微错开些，当你翻阅备忘录时，效果就像动画一样。我们将在应用 BubbleDrawGUI 中添加这种效果，让气泡看起来四处漂浮。

为创建动画，需要在一秒钟内多次这样做：在屏幕上绘制所有的气泡，稍微调整气泡的位置，再重绘屏幕。绘制的每个画面都是一**帧**。如果重绘速度足够快，我们的眼睛和大脑将填补相邻帧之间的空白，让我们以为物体在平稳地移动。动画的**帧速**（重绘屏幕的速度）通常为每秒大约 30 帧。我们将使用一个新类——创建定时器的 `javax.swing.Timer`——来告诉程序何时该重绘气泡。我们还将使用一个事件处理程序，在定时器到期后都更新气泡的位置并重绘屏幕。

要创建动画式气泡，需要完成 4 步：添加定时器、设置定时器、准备动画以及启动定时器。使用定时器在其他 Java 游戏或应用中添加动画时，也将采取这 4 个步骤。

10.2.1 添加定时器

要在应用中添加定时器，需要导入 `javax.swing.Timer` 类。为此，在文件 BubblePanel.java 开头添加如下 `import` 语句：


```
import javax.swing.Timer;
import java.awt.event.*;
import java.util.ArrayList;
import java.awt.Graphics;
import java.util.Random;
```

```
import java.awt.Color;
import javax.swing.JPanel;
```

导入 javax.swing 包中的 Timer 类后，就可创建定时器对象并随便指定触发频率。请注意，在上述代码片段中的第 2 行，我们导入了 java.awt.event.*。这行代码导入所有的 java.awt 事件处理程序，包括我们将用来处理定时器事件的 ActionListener 类。

接下来，在 BubblePanel 类中添加两个变量：用于存储定时器本身的变量 timer；用于存储定时器等待多少毫秒后重绘屏幕的 int 变量 delay：

```
public class BubblePanel extends JPanel {
    Random rand = new Random();
    ArrayList<Bubble> bubbleList;
    int size = 25;
    Timer timer;
    int delay = 33;
```

在 Java 中，定时器需要知道等待多少毫秒后触发定时器事件。1 毫秒真的非常短，因此我将延迟设置成了 33 毫秒。这将导致每秒大约重绘屏幕 30 次，因为 1 秒 = 1000 毫秒，而 1000 / 33 = 30。这样的速度大致与电视上的卡通片相当。

10.2.2　设置定时器

现在可以设置定时器了。在构造函数 BubblePanel()中添加如下代码行，以初始化定时器并设置指定的延迟：

```
    public BubblePanel() {
        timer = new Timer(delay, new BubbleListener() );
        bubbleList = new ArrayList<Bubble>();
        setBackground(Color.BLACK);
        // testBubbles();
        addMouseListener( new BubbleListener() );
        addMouseMotionListener( new BubbleListener() );
        addMouseWheelListener( new BubbleListener() );
    }
```

创建定时器的构造函数 Timer()接受两个参数：第一个是以毫秒为单位的延迟，第二个是监听定时器事件的事件处理程序。定时器到期后触发 actionPerformed()事件，这种事件类似于按钮单击事件 actionPerformed()。定时器有点像每隔一段时间就单击自己的按钮。我们在构造函数中创建定时器，这样可在 GUI 事件发生时修改它。

为了监听定时器事件，我们将修改 BubbleListener 类。请在文件 BubblePanel.java 中向下滚动，找到前面创建的私有类 BubbleListener，并在这个类的左大括号前面加上 implements ActionListener：

```
private class BubbleListener extends MouseAdapter implements ActionListener {
```

这项修改让 BubbleListener 类实现 java.awt.event.*中的 ActionListener 类,以便能够监听 actionPerformed()事件。实现事件监听器类是另一种处理用户事件的方式。为了处理定时器事件,需要添加一个 actionPerformed()事件处理程序。为此，在 BubbleListener 类末尾添加如下方法：

```
private class BubbleListener extends MouseAdapter implements ActionListener {
    public void mousePressed(MouseEvent e) {
        --snip--
    }
    public void mouseDragged(MouseEvent e) {
        --snip--
    }
    public void mouseWheelMoved(MouseWheelEvent e) {
        --snip--
    }
    public void actionPerformed(ActionEvent e) {
    }
}
```

我们将在这个新增的 actionPerformed()方法中添加代码，在定时器到期时移动气泡并重绘屏幕。下面就来这样做。

10.2.3 准备动画

添加定时器，并让 BubbleListener 监听定时器事件后，需要告诉 Java 在定时器触发事件时如何做。

每当定时器触发事件时，就应绘制气泡动画序列的下一帧。为此，我们将首先让事件处理程序 actionPerformed()更新气泡并重绘屏幕,然后让 Bubble 类更新气泡——在屏幕上向上移动它。我们要让程序每秒执行这些步骤约 30 次。

在前面给 BubbleListener 类添加的方法 actionPerformed()中，添加下面 3 行代码以更新气泡并重绘屏幕：

```
    public void actionPerformed(ActionEvent e) {
     ❶ for (Bubble b : bubbleList)
         ❷  b.update();
     ❸ repaint();
    }
```

在❶处,我们使用 for-each 版的 for 循环遍历 bubbleList 中的每个气泡 b。别忘了,bubbleList 是一个 ArrayList，包含用户通过单击和拖曳创建的所有气泡。

在遍历 bubbleList 中每个气泡的过程中，我们对所有的气泡都调用新方法 update()❷。Eclipse 给这条语句加上了红色下划线，因为我们还没有在 Bubble 类中定义方法 update()，但我们马上就会这样做。我们将在方法 update()中修改气泡的位置,让它们看起来像不断往屏幕顶部飘。

在❸处，我们调用方法 repaint()刷新屏幕——清空绘图窗口并在更新后的新位置处绘制气泡。通过每秒这样做 30 次，可实现所需的动画效果。

下面来创建方法 update()，让 Bubble 类知道在定时器触发时如何移动气泡。在 Java 的(x, y)坐标系中，屏幕顶部的 y 坐标为 0，因此需要减小 y 值。为了让气泡看起来像在向上移动，可在每次更新时都将 y 坐标减去一个较小的值。

请向下滚动到文件 BubblePanel.java 末尾（定义 Bubble 类的地方），并在方法 draw()后面添加方法 update()，如下所示：

```
        public void draw(Graphics canvas) {
            canvas.setColor(color);
            canvas.fillOval(x - size/2, y - size/2, size, size);
        }
        public void update() {
            y -=5;
        }
    }
}
```

为了更新气泡的位置，这个方法将气泡的 y 值减去 5 个像素。每次重绘气泡时，它都将向上移动 5 个像素。

将文件存盘。再完成一步就可以运行这个应用了！

10.2.4　启动定时器

为在应用 BubbleDrawGUI 中添加动画，需要做的最后一步是启动定时器。请向上滚动到构造函数 BubblePanel()处，并在其中添加如下代码行：

```
    public BubblePanel() {
        timer = new Timer(delay, new BubbleListener() );
        --snip--
        addMouseWheelListener( new BubbleListener() );
        timer.start();
    }
```

方法 timer.start()启动定时器，使其每隔指定的毫秒数就触发事件，直到你调用方法 timer.stop()或退出程序。

请保存并运行程序。当你绘制气泡时，它们将平稳地向上飘，这都是拜定时器事件处理程序所赐。

现在不仅有令人着迷的动画效果，第 9 章实现的鼠标滑轮滚动和其他功能也依然管用。当前，所有气泡都沿一个方向飘，但我们已营造出了所需的移动假象。在下一节，你将学习如何让气泡四处飘动。

10.3　随机选择速度和方向

前一节创建的方法 update()只修改了每个气泡的 y 坐标，导致每次重绘屏幕时所有气泡都垂

直向上移动。在本节中，我们将让气泡沿垂直和水平方向以随机的速度移动，让它们看起来像四处飘散，如图 10-4 所示。

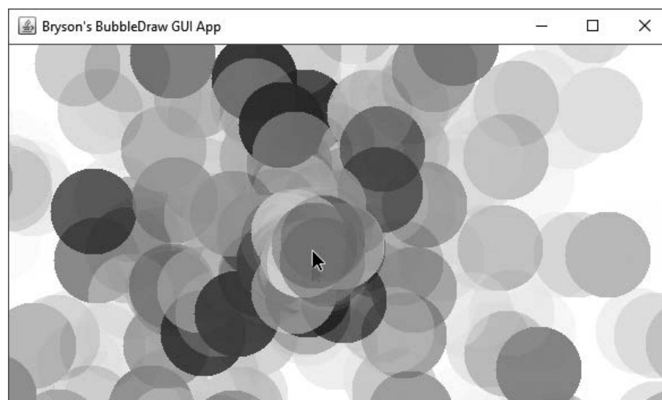


图 10-4　修改每个气泡的 *x* 和 *y* 坐标，让气泡看起来像是四处飘散

气泡的水平速度指的是每帧向左或向右移动多少像素，这决定了气泡的新 *x* 坐标。同理，气泡的垂直速度决定了其新的 *y* 坐标。只需同时让气泡沿水平和垂直方向移动，就可让它沿任何方向移动。图 10-5 表明，水平和垂直速度一起营造了气泡沿斜线方向移动的假象。

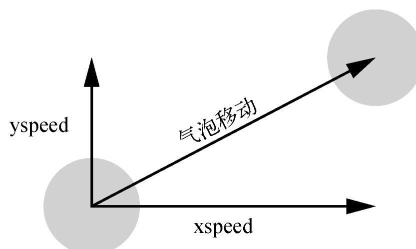


图 10-5　通过快速修改气泡的 *x* 和 *y* 坐标，让气泡看起来像是沿斜线方向移动

首先添加一些变量，用于存储每次重绘屏幕时气泡都将沿水平和垂直方向移动的像素数。请在 Bubble 类开头添加如下两行代码：

```
private class Bubble {
    --snip--
    private Color color;
  ❶ private int xspeed, yspeed;
  ❷ private final int MAX_SPEED = 5;
```


在❶处，我们声明了两个整型变量——xspeed 和 yspeed，其中前者用于存储每次更新屏幕时气泡将沿水平方向移动的像素数，而 yspeed 用于存储气泡沿垂直方向移动的像素数。在❷处，我们添加了一个**常量**——MAX_SPEED，用于存储气泡每次移动的最大像素数。常量是具名值，类

似于变量，但在程序中保持不变，因此我们将其声明为 final 的，让 Java 知道常量的值是固定的。作为约定，常量名为全大写，以便能够将其与变量区分开来。

与颜色一样，我们将使用方法 rand.nextInt()给每个气泡指定随机的水平和垂直速度。为此，需在构造函数 Bubble()中添加如下两行代码：

```
public Bubble(int newX, int newY, int newSize) {
    x = newX;
    y = newY;
    size = newSize;
    color = new Color(rand.nextInt(256),
            rand.nextInt(256),
            rand.nextInt(256),
            rand.nextInt(256) );
    xspeed = rand.nextInt(MAX_SPEED * 2 + 1) - MAX_SPEED;
    yspeed = rand.nextInt(MAX_SPEED * 2 + 1) - MAX_SPEED;
}
```

我们要让气泡能够沿任何方向移动，因此需要水平速度和垂直速度，但只需两个变量就可表示 4 个方向（左右和上下）。为此，我们让水平速度和垂直速度可正可负。xspeed 为负时气泡将向左移动，为正时气泡将向右移动。yspeed 为负时气泡将向上移动，为正时气泡将向下移动。我将 MAX_SPEED 乘以 2 再加上 1，结果为 11（5 * 2 + 1 = 11），因此 rand.nextInt(MAX_SPEED * 2 + 1)与 rand.nextInt（11）等价——生成一个 0~10 的随机数。通过将生成的随机数减去 MAX_SPEED，结果将在范围-5 和+5 之间（因为 0 - 5 = -5，而 10 - 5 = 5），这让 xspeed 和 yspeed 可正可负。

最后，我们需要修改方法 update()，在每次重绘屏幕时都将气泡移到新位置。请将语句 y -= 5;替换为如下两条语句：

```
public void update() {
    x += xspeed;
    y += yspeed;
}
```

在这里，每次重绘屏幕时，都将每个气泡沿水平和垂直方向分别移动 xspeed 和 yspeed（这两个值是为气泡随机生成的），而不是向上移动 5 个像素。其结果是，色彩缤纷的气泡向四处移动。

完成这些修改后，保存并运行程序，你将看到类似于图 10-4 所示的动画效果。通过沿水平和垂直方向移动气泡，营造出了气泡的移动速度和方向都是随机的假象。

你可能注意到了一种怪异的情况，那就是有些气泡呆在原地不动，如图 10-6 中央的那些气泡。这是因为水平速度和垂直速度为−5 到+5 的随机数字，而有些气泡的 xspeed 和 yspeed 都为零，它们根本不会移动。

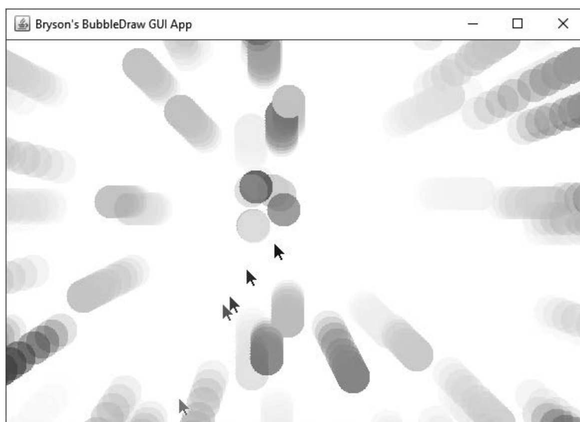


图 10-6　由于随机速度值可能为零，因此有些气泡可能呆在原地不动，如图中中央附近的那些气泡

为避免气泡呆在原地不动，可检查 xspeed 和 yspeed 是否都为零；如果都为零，就修改其中一个或两个。这是本章末尾的编程练习 1 要求你完成的任务。

让气泡四处飘散后，你可能还想给应用添加两项功能：暂停动画和清屏。现在该给这个动画式图形应用创建 GUI 了。

10.4　为应用创建 GUI

应用 BubbleDrawGUI 是图形化的，但没有类似于其他 GUI 应用的界面。通过添加一个 Pause/Start 按钮和一个 Clear 按钮（如图 10-7 所示），用户将更容易理解这个应用，与之交互起来也更简单。下面就来添加这些按钮。



图 10-7　Pause/Start 和 Clear 按钮

10.4.1　添加面板和按钮

在 Package Explorer 面板中，右击文件 BubblePanel.java 并选择 Open With▶WindowBuilder Editor。单击选项卡 Design，你将看到 GUI 设计视图。

首先，我们来添加一个 JPanel，作为按钮 Pause/Start 和 Clear（以及你以后可能要添加的其他 GUI 组件）的容器。在 Palette 面板中，选择 Containers 中的 JPanel，然后将鼠标指向设计预览并单击 BubblePanel 设计预览，将 JPanel 放在黑色背景上。

也可这样添加 JPanel：在选择了 JPanel 组件的情况下，单击面板 Structure 下方的 Components 面板中的 javax.swing.JPanel。你将在黑色的 BubblePanel 设计预览顶部看到一个非常小的灰色 JPanel。

接下来添加 Pause/Start 和 Clear 按钮。在 Palette 面板中，向下滚动到 Components 部分，选择组件 JButton，再将鼠标指向刚才添加的小型 JPanel 并单击，以添加第一个 JButton。将这个按钮的 text 属性设置为 Pause，为此可直接在 GUI 预览中输入，也可在 Properties 面板中设置（稍后你将明白我们为何称之为 Pause/Start 按钮）。

采用同样的步骤添加 Clear 按钮：在 Palette 面板中选择 JButton，在 JPanel 中单击以添加按钮，再将按钮的 text 属性设置为 Clear。

如果 JPanel 太小，难以在其中添加按钮，可在 Palette 面板中选择 JButton，再单击 Structure 面板中的 panel，将按钮放在这个面板内，如图 10-8 所示。将这两个按钮分别命名为 btnPause 和 btnClear，为此可在添加按钮时这样做，也可在 Properties 面板中修改属性 Variable。

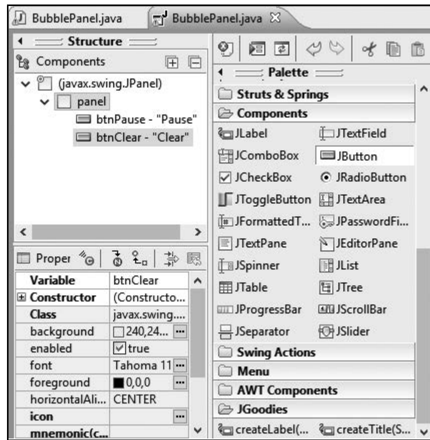


图 10-8　创建 GUI 时，可在 Palette 面板中选择组件，并将它们直接添加到 Structure 面板中。在这里，我们将 btnPause 和 btnClear 添加到刚创建的面板中

无法看清设计预览中的 JPanel，或者要修改 GUI 中组件的排列顺序和编组方式时，Structure 面板提供了将组件添加到 GUI 中的方便途径。图 10-9 显示了设计好的 GUI 中的两个按钮。



图 10-9　最终的 GUI 中的 Pause/Start 和 Clear 按钮

在绘图屏幕顶端添加按钮 Pause/Start 和 Clear 后，该给这些按钮编写事件处理程序了。

10.4.2 给按钮 Clear 和 Pause/Start 编写事件处理程序

我们先从按钮 Clear 着手。要清除屏幕上所有的气泡，一种办法是将变量 bubbleList 重置为空列表，这样将没有要绘制的气泡，而用户可着手重新绘制。为实现这种行为，请双击按钮 Clear（别忘了，在 Design 选项卡中双击按钮将为其创建一个事件监听器，并切换到 Source 选项卡），再在 btnClear 的操作监听器的大括号内添加如下两行代码：

```
JButton btnClear = new JButton("Clear");
btnClear.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
     ❶ bubbleList = new ArrayList<Bubble>();
     ❷ repaint();
    }
});
panel.add(btnClear);
```

在❶处，我们清空变量 bubbleList：将其设置为一个存储 Bubble 对象的新 ArrayList。这个新列表是空的，因此我们只需重绘屏幕就能得到一个干净的黑色背景，就像应用刚启动时一样。在❷处，我们调用函数 repaint()绘制出新的空屏幕。

保存并运行应用，再绘制几个气泡并单击 Clear 按钮清屏。

切换到 Design 选项卡，并双击 Pause/Start 按钮再创建一个事件监听器。用户单击 Pause/Start 按钮时，我们不仅要通过停止定时器来停止动画，还要将这个按钮的文本改为 Start。而当用户再次单击这个按钮时，我们要重启定时器以恢复动画，并将该按钮的文本改回为 Pause。

在你双击 Pause/Start 按钮时 Eclipse 提供的方法 actionPerformed()中，输入下面的代码。

注意 确保方法 actionPerformed()的 ActionEvent 参数名为 e，下面的代码使用粗体突出了这一点。

```
JButton btnPause = new JButton("Pause");
btnPause.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
     ❶ JButton btn = (JButton)e.getSource();
     ❷ if (btn.getText().equals("Pause")) {
         ❸ timer.stop();
         ❹ btn.setText("Start");
        }
        else {
         ❺ timer.start();
         ❻ btn.setText("Pause");
        }
    }
});
panel.add(btnPause);
```


在❶处，我们使用方法 e.getSource()来确定单击的是哪个按钮，将其转换为类型 JButton，再将结果存储到变量 btn 中。需要确定特定的 GUI 元素是否被单击或修改时，方法 getSource()

很有用，在你同时为多个元素编写事件处理程序时尤其如此。在这个示例中，我们可使用 getSource()来访问按钮的属性，如 text 属性。

在❷处，我们检查按钮上的文本是否为字符串"Pause"；如果是，就停止定时器❸以暂停动画，再将按钮的文本修改为字符串"Start"❹。

如果按钮上的文本不是"Pause"（换而言之，如果动画已暂停，而按钮上的文本因前一次单击而被改为"Start"），这个事件处理程序将执行 else 语句：启动定时器❺以恢复动画，并将按钮上的文本改回为"Pause"❻。

保存这个文件并再次运行程序。现在你可暂停动画、绘制气泡再重启动画，从而实现令人瞠目结舌的气泡爆炸效果，如图 10-10 所示。

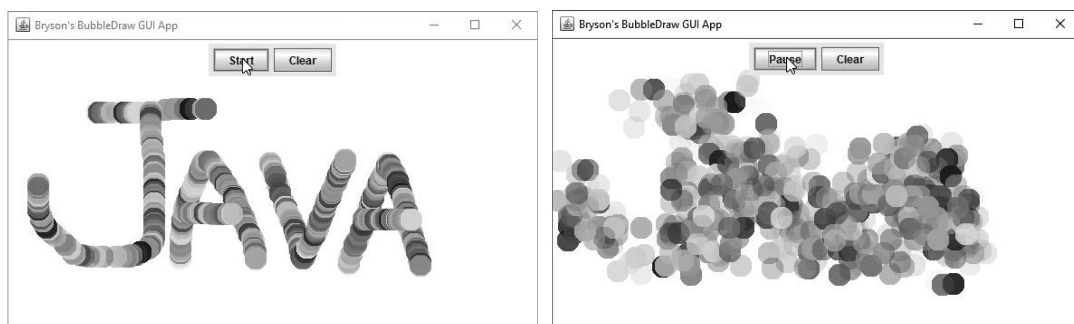


图 10-10　暂停动画、绘制一个形状再重启动画，绘制的图形将爆炸成色彩缤纷的气泡

BubbleDrawGUI 是一个极具视觉震撼力的动画应用，而其中的按钮赋予了用户更大的画面控制权。但动画开始后，气泡将逐渐飘到屏幕边缘外面，一去不复返。如果要让气泡在窗口内弹回来弹回去，而不飘到屏幕外面，该如何做呢？

10.5　使用碰撞检测让气泡到达窗口边缘后往回弹

动画并非只能用于实现翻书效果和屏保，它在计算机游戏中也无处不在。无论是在移动应用还是最新的在线或控制台游戏中，动画都是游戏开发人员让用户感觉到运动和动作的途径。

在应用 BubbleDrawGUI 中，可使用的一个很有用的游戏编程概念是**碰撞检测**，它让我们能够检查屏幕上的两个物体是否发生了重叠或**碰撞**。在视频游戏中，可在玩家向敌方的太空飞船射击或踢球时，使用碰撞检测来确定程序该如何做。在这个应用中，我们要确定气泡是否到达绘画屏幕的边缘，以修改气泡的移动方向，使其看起来像是从屏幕边缘弹回中央一样，如图 10-11 所示。

通过使用碰撞检测，可让虚拟物体（如 BubbleDrawGUI 中的气泡）更逼真。在你喜欢的计算机游戏中，正是碰撞检测避免了玩家跌到地板下面或穿墙而过。这些物体是虚构的，不过是计算机图形而已，因此并不会真的发生碰撞，但碰撞检测可营造出它们很坚固的假象。因此，如果让气泡到达屏幕边缘时缓慢弹回，将让用户觉得它们更像真实的物体。下面就来看看如何使用碰撞检测。



图 10-11　气泡从窗口边缘往回弹，这多亏了碰撞检测

10.5.1　软性回弹

首先，我们将详细介绍碰撞检测如何起作用，让气泡到达窗口边缘后回弹。我们知道，每个气泡都有表示其位置的 *x* 和 *y* 坐标，还有 xspeed 和 yspeed，它们分别表示每次刷新屏幕时，气泡将沿水平和垂直方向移动多少个像素。

要确定气泡是否碰到了窗口边缘，需要知道屏幕边缘的 *x* 和 *y* 坐标，这样才能将其与气泡的 *x* 和 *y* 坐标进行比较。在屏幕左边缘，*x* 坐标是最小的，即 x==0；在屏幕上边缘，*y* 坐标是最小的，即 y==0。但屏幕的右边缘和下边缘呢？

在 Java 中，每个 GUI 组件都继承了两个方法——getWidth()和 getHeight()，它们分别返回组件的宽度和高度。在应用 BubbleDrawGUI 中，绘图屏幕为 JPanel BubblePanel，因此如果我们在 BubblePanel 中调用方法 getWidth()和 getHeight()，它们返回的值将分别为最大的 x 和 y 值。

我们在 Bubble 类的方法 update()中检查气泡的 x 和 y 值，以判断它是否碰到了屏幕边缘。你可能还记得，在方法 update()中，我们还根据 xspeed 和 yspeed 修改气泡的 *x* 和 *y* 坐标，以营造气泡在运动的假象。

我们来更精确地定义“回弹”的含义。在图 10-11 中，气泡到达屏幕的右边缘，而右边缘的 x 值为 getWidth()，这意味着气泡的 x 值为最大的 *x* 坐标——以像素为单位的屏幕宽度。为了让气泡看起来像是被弹回，我们修改其移动方向——对 xspeed 取负。之前，气泡每次更新时移动的像素数都为正；等气泡碰到屏幕右边缘后，我们可对 xspeed 取负，让气泡沿相反的方向移动。换而言之，我们可让 xspeed 变成负的，让气泡向左移动，从而离屏幕右边缘越来越远。

要对气泡的水平移动速度取负，可将 xspeed 设置为-xspeed，这将反转 xspeed。也就是说，如果 xspeed 为每帧 3 像素，气泡碰到屏幕右边缘后将往回弹，导致 xspeed 变成每帧–3 像素。

对于屏幕的左边缘（x==0），可做同样的处理。如果气泡的 x 值导致气泡碰到了左边缘，就将 xspeed 设置为-xspeed，让水平移动方向反转：如果 xspeed 原来为–3，它将变成–(–3)，即+3。这将让气泡向右移动，离屏幕左边缘越来越远。

在文件 BubblePanel.java 中，向下滚动到末尾——定义 Bubble 类的地方。找到方法 update()

并添加如下代码，检测气泡是否碰到了屏幕左边缘或右边缘：

```
public void update() {
    x += xspeed;
    y += yspeed;
    if (x <= 0 || x >= getWidth())
        xspeed = -xspeed;
}
```

如果气泡的 x 值小于等于零或大于等于屏幕的宽度，气泡必然碰上了屏幕的左边缘或右边缘，因此我们修改 xspeed，让气泡回弹——沿相反的方向移动。

对于上边缘和下边缘，我们采取同样的处理办法，但修改 yspeed：

```
public void update() {
    x += xspeed;
    y += yspeed;
    if (x <= 0 || x >= getWidth())
        xspeed = -xspeed;
    if (y <= 0 || y >= getHeight())
        yspeed = -yspeed;
}
```

如果气泡的 y 值小于等于零或大于等于单位为像素的屏幕高度，我们就将 yspeed 设置为 -yspeed，让气泡沿相反的方向移动。

在方法 update()中添加这些代码后，保存并运行程序。这次气泡碰到屏幕边缘后将往回弹。然而，你可能会发现一个小小的问题：无论气泡沿什么方向移动，都要等到有一半在屏幕外面后才往回弹，如图 10-12 所示。

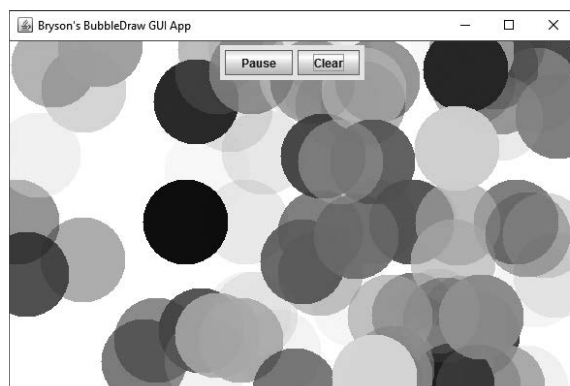


图 10-12　窗口边缘上有很多气泡，它们要等到几乎有一半在外屏幕外面后才往回弹

之所以会出现这种“软性”回弹，是因为我们检测的是气泡**中心**是否碰到了屏幕边缘。在第 9 章，我们让气泡的中心位于用户单击的(x, y)坐标处，因此每个气泡的 x 和 y 值表示的都是该气泡的中心位置。为了让气泡完全在屏幕内，需要检测气泡的外边缘是否碰到了绘图窗口边缘。

10.5.2　硬性回弹

为了检查气泡边缘是否碰到了屏幕边缘，必须将气泡中心到气泡边缘的距离考虑在内，这个距离就是气泡的半径（因为每个气泡都是圆形的）。气泡的半径就是气泡尺寸的一半，即 size/2。为将气泡的尺寸考虑在内，请像下面这样修改方法 update()中的两条 if 语句：

```
public void update() {
    x += xspeed;
    y += yspeed;
❶  if (x - size/2 <= 0 || x + size/2 >= getWidth())
        xspeed = -xspeed;
❷  if (y - size/2 <= 0 || y + size/2 >= getHeight())
        yspeed = -yspeed;
}
```

在❶处，我们将 x 将去 size/2，以判断气泡左边缘是否碰到了屏幕左边缘。如果 x - size/2 小于等于零，就碰到了。除法的优先级高于减法，因此将先计算 size/2，再将 x 减去得到的商，所以无需将 size/2 放在括号内。另外，我们将 x 加上 size/2，以判断气泡右边缘是否碰到了屏幕右边缘。如果 x + size/2 大于等于 getWidth()，就碰到了。在❷处，我们做了同样的修改，以判断气泡的上边缘（y - size/2）和下边缘（y + size/2）是否碰到了绘图窗口的上边缘和下边缘。

保存并再次运行程序。现在所有的气泡（无论大小）都在碰到窗口边缘后立刻回弹，如图 10-13 所示。

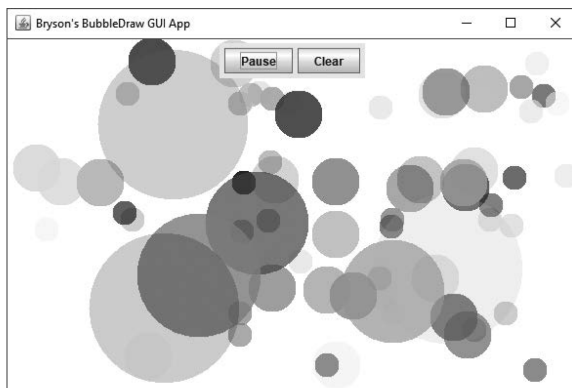


图 10-13　现在所有气泡都显得更坚固，碰到窗口边缘后就立刻回弹（硬性回弹）

请单击应用标题栏中的最大化按钮或双击标题栏，以扩大窗口。你将看到，应用处于全屏模式时，气泡将移动更远的距离，到达绘图窗口边缘后才往回弹。这是因为使用了方法 getWidth() 和 getHeight()来确定右边缘和下边缘的位置，而这两个方法总是返回当前宽度和高度，因此你可在绘画期间随便调整应用窗口的尺寸。

下面来添加最后一项功能，通过 GUI 赋予用户更大的控制权。

10.6　添加用于控制动画速度的滑条

当前，用户能够暂停动画和清屏，还能创建大小各异的气泡。下面来让用户能够控制动画的速度——提供一个修改定时器延迟的滑条，如图 10-14 所示。

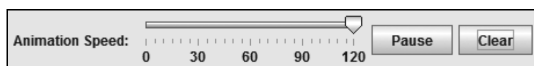


图 10-14　添加一个滑条，让用户能够提高或降低动画的速度

首先，切换到 Design 选项卡，并在 GUI 控件面板中添加一个 `JLabel` 和一个 `JSlider`。为此，在面板 Palette 中向下滚动到 Components 部分，并选择其中的 JLabel。在 GUI 设计预览中，在小面板中的 Pause/Start 按钮前面单击，将标签放到这里，再将标签的文本改为 `Animation Speed:`。

接下来，单击面板 Palette 中的 JSlider 组件，再在标签 Animation Speed 和按钮 Pause/Start 之间单击，将滑条放到这里，如图 10-15 所示。

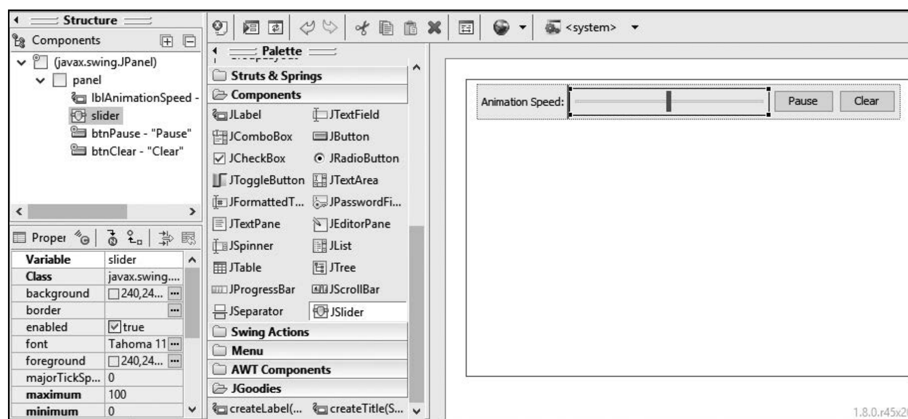


图 10-15　在设计视图中的 GUI 控件面板中添加一个标签和一个滑条

你可能注意到了，当我们添加元素时，用于放置所有 GUI 控件的 `JPanel` 将相应地增大。如果你在 Structure 面板中单击这个 `JPanel`，将发现其 `Layout` 属性为默认值 `java.awt.FlowLayout`。这种布局让 `JPanel` 自动增大，以容纳你加入到其中的所有 GUI 元素。在猜数游戏和应用 Secret Messages 中，我们使用了布局 `AbsoluteLayout`，因为我们要准确地指定元素的位置。在这个绘画应用中，可以更灵活；为方便随时添加 GUI 组件，布局 `FlowLayout` 最为理想。

10.6.1　定制滑条

我们将修改几个属性以定制滑条，为此先得确定要让它是什么样的。这个滑条应该让用户能够轻松而直观地修改动画速度，换而言之，如果用户将滑条移到最左边，动画速度应降低到几乎停止的状态，而如果用户将滑条移到最右边，动画的速度应非常快。

显示器刷新屏幕的速度通常为 30~120 次每秒，但最常见的是 60 Hz（频率单位赫兹）。如果动画速度超过了 120 次每秒，显示器可能无法显示所有的动画帧，因此合理的做法是，将这个滑条的最大值设置为 120 帧每秒。

每秒的帧数（fps）是一个度量动画平滑程度的指标。在计算机上，60 fps 的游戏看起来比 30 fps 的游戏更顺畅。

请选择设计预览中的滑条，再在左下角的 Properties 面板中设置取值范围：将属性 `maximum` 设置为 120，并将属性 `minimum` 设置为 0（默认值）。为设置标签和刻度，将 `majorTickSpacing` 和 `minorTickSpacing` 分别设置为 30 和 5，再选择 `paintLabels`、`paintTicks` 和 `paintTrack` 旁边的复选框，将这些属性都设置为 `true`。最后，将属性 `value` 设置为 30，这指定了默认的 fps（每秒帧数）值。图 10-16 显示了 Properties 面板中的自定义值以及修改后的 `JSlider`。

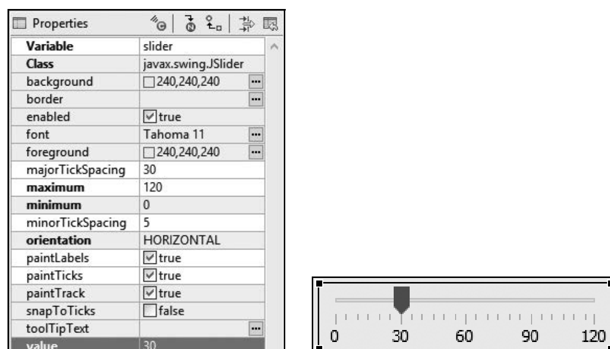


图 10-16 Properties 面板中滑条 Animation Speed 的自定义属性值（左）以及定制后的滑条 Animation Speed（右）

在这个滑条上，以间隔 30 的方式标出了 0~120 的值，这是因为你选择了复选框 `paintLabels`，并将 `majorTickSpacing` 设置成了 30。在这些值之间，还显示了小刻度，因为你将 `minorTickSpacing` 设置成了 5，并选择了复选框 `paintTicks`。为方便修改动画速度定制滑条后，下面来编辑代码让滑条发挥作用。

10.6.2 实现滑条事件处理程序

在 Design 选项卡的设计预览中，右击滑条并选择 Add event handler▶change▶stateChanged，Eclipse 将添加一个包含方法 `stateChanged()`的 `ChangeListener`，它类似于我们在应用 Secret Messages 的滑条实现中使用的 `ChangeListener`，如下所示：


```
JSlider slider = new JSlider();
slider.addChangeListener(new ChangeListener() {
    public void stateChanged(ChangeEvent e) {
    }
});
```

首先，我们需要在类开头声明一个 JSlider 变量，以便能够在事件处理程序 stateChanged() 的代码中访问这个滑条。为此，向上滚动到 BubblePanel 类的开头，并在变量 timer 和 delay 后面添加如下代码行：

```
public class BubblePanel extends JPanel {
    Random rand = new Random();
    ArrayList<Bubble> bubbleList;
    int size = 25;
    Timer timer;
    int delay = 33;
    JSlider slider;
```

接下来，向下滚动到滑条的代码处，并删除第一行开头的 JSlider 类型声明：

```
        slider = new JSlider();// 删除行首的 JSlider
        slider.addChangeListener(new ChangeListener() {
            public void stateChanged(ChangeEvent e) {
            }
        });
```

我们要在用户调整滑条位置时修改动画速度——修改定时器事件之间的延迟。为此，需要从滑条获取速度值，将其转换为毫秒数，再将定时器延迟设置为这个新值。请在方法 stateChanged() 的大括号内添加如下代码行：

```
        slider = new JSlider();
        slider.addChangeListener(new ChangeListener() {
            public void stateChanged(ChangeEvent arg0) {
              ❶ int speed = slider.getValue() + 1;
              ❷ int delay = 1000 / speed;
              ❸ timer.setDelay(delay);
            }
        });
```

在❶处，我们使用方法 getValue()从滑条获取速度值，并将其存储在整型变量 speed 中。请注意，将滑条的值加上了 1，这样做旨在避免❷处将 1000 除以 speed 以计算帧间延迟（毫秒数）时发生被零除错误。滑条的值可降低为零，但通过将滑条值加上 1，可防止计算延迟时引发错误：1000/0 引发除零异常，而 1000/1 导致帧间延迟非常长（1000 毫秒），让人觉得动画像是停止不动的。这意味着当用户将滑条移到 0 处时，动画实际上并不会停止。要真的停止动画，用户必须单击 Pause/Start 按钮。

最后，我们通过设置延迟值（单位为毫秒）❸来更新定时器。这将修改相邻定时器事件之间的时间，从而提高或降低动画速度。

保存这个文件并运行程序，将滑块左右移动，你将发现你现在能够控制动画的速度了。

在我们开发的应用中，BubbleDrawGUI 是交互性最强、最有趣、视觉冲击力最强的。这是一个动画式绘图应用，其 GUI 让用户能够完全控制动画。请玩一会儿，并祝贺自己做得非常出色！

分享应用

这个应用也很出色，可与朋友分享。为了从 Eclipse 导出可执行的 JAR 文件，请选择菜单 File▸Export，再展开文件夹 Java 并单击 Runnable JAR file。单击 Next 按钮，再单击 Launch configuration 部分的下拉列表并选择 BubbleDrawGUI – BubbleDrawGUI.

在 Export destination 部分，单击 Browse 按钮，再选择要将文件导入到哪个文件夹（可以是“桌面”）。给文件指定名称，如 Bryson’s BubbleDraw.jar，再依次单击 Save 和 Finish 按钮。

找到你保存的 JAR 文件，运行它，再与朋友分享。即便你的朋友没有安装 Eclipse，但只要安装了 Java，他也能运行应用 BubbleDrawGUI。

10.7 小结

本章以第 9 章创建的 BubbleDraw 应用为基础，创建了气泡能够回弹的动画版。下面是你在本章学到的一些技能：

- 给图形应用添加 GUI；
- 在 Eclipse 的 Package Explorer 中通过复制并粘贴来创建项目的副本；
- 通过重构来重命名类或对象；
- 设置 RGB 颜色的 alpha 组分以指定透明度；
- 创建、设置并启动 `Timer` 对象；
- 处理 `Timer` 对象触发的事件；
- 使用定时器来移动图形对象，以创建动画；
- 使用碰撞检测让虚拟物体回弹；
- 使用 `getWidth()`和 `getHeight()`来确定窗口边缘的位置；
- 使用滑条修改定时器的延迟；
- 修改定时器的 `delay` 属性以调整动画的速度。

10.8 编程练习

为复习并使用学到的知识以及获得更多的编程技能，请尝试完成这里的编程练习。如果遇到困难，可从本书配套网站（https://www.nostarch.com/learnjava/）下载示例解决方案。

10.8.1 编程练习 1：避免气泡呆在原地不动

在本章前面，我们注意到了一个问题，那就是对于有些气泡，随机生成的速度为零，导致它们在其他气泡漂走时呆在原地不同。如果气泡的 `xspeed` 和 `yspeed` 都为零，它就会呆在原地不动。在这个编程练习中，你将添加一些代码，避免给气泡生成的随机速度为零。为此，你需要检查

xspeed 和 yspeed，看它们是否都为零；如果是这样的，就将它们设置为其他值，如 1。

提示　请在构造函数 Bubble()中添加这条 if 语句，并将其放在设置 xspeed 和 yspeed 的代码后面。

```
private class Bubble {
    --snip--
        public Bubble(int newX, int newY, int newSize) {
        --snip--
            xspeed = rand.nextInt(MAX_SPEED * 2) - MAX_SPEED;
            yspeed = rand.nextInt(MAX_SPEED * 2) - MAX_SPEED;
            if // 在这里添加代码
        }
```

修改后存盘并运行应用，你将发现再也没有气泡呆在原地不动。

10.8.2　编程练习 2：创建应用 FlexiDraw

由于速度是随机的，因此气泡四处飘散，这正是我们最初要实现的效果。但如果我们要绘制形状，并让它们一起移动，该如何做呢？

通过将所有气泡的速度都设置为相同的值，可在气泡从屏幕边缘回弹时形成有趣的扭曲效果，如图 10-17 所示。



图 10-17　通过将所有气泡的 xspeed 和 yspeed 都设置为相同的值，可创建扭曲的回弹效果：气泡一起移动，到达屏幕边缘时扭曲并回弹

在这个编程练习中，你将创建本章应用的副本，以保留原来的版本不变。为此，在 Package Explorer 面板中，复制并粘贴项目 BubbleDrawGUI，将其重命名为 FlexiDraw 或你选择的其他名称。在文件 BubblePanel.java 中，修改构造函数 Bubble()，使其不给变量 xspeed 和 yspeed 设置随机值，而是将它们设置为相同的值，如下所示：

```
        xspeed = yspeed = 2;
```

这行代码利用了赋值运算符(=)的一个有趣的性质。这被称为**串接赋值**，因为 xspeed 和 yspeed 都被赋值为 2，而等号让我们能够一次性将相同的值赋给多个变量。

你可将 xspeed 和 yspeed 设置为比这里更大或更小的值。这里的重点是，我们没有使用随机

速度，而是让所有气泡的初始移动方向和速度都相同。每个气泡都在每次重绘时右移 2 像素并下移 2 像素，因此所有的气泡将步调一致地移动。

将这个文件存盘，再在 Package Explorer 中右击文件夹 FlexiDraw，并选择 Run As▶Java Application。暂停动画并绘制一些气泡，然后单击 Start 按钮，你将发现气泡到达屏幕边缘后，它们组成的形状将变形、扭曲并回弹。你还可在不停止动画的情况下绘制气泡，这样将出现很酷的螺旋效果。

10.8.3　编程练习 3：PixelDraw 2.0

在这个编程练习中，将重用你在第 9 章的编程练习 2 中编写的代码。通过在本章的动画式绘画应用中添加像素化效果，可绘制以动画方式移动的方块形状。再加上前一个练习实现的功能，可在应用中实现类似于 Minecraft 的回弹和扭曲效果，如图 10-18 所示。

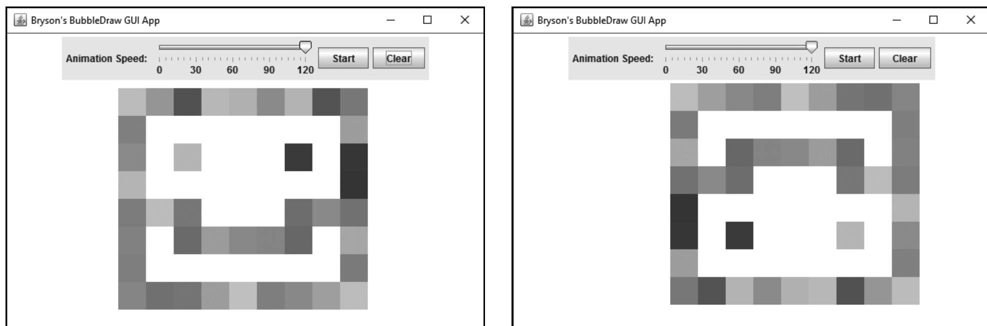


图 10-18　PixelDraw 2.0 实现的效果（左）；到达边缘后回弹并扭曲，形状完全颠倒过来了（右）

在暂停动画的情况下绘画，可让方块构成整洁的网格形状，如图 10-18 所示。在不停止动画的情况下绘画，可实现堆叠式 3D 效果，如图 10-19 所示。

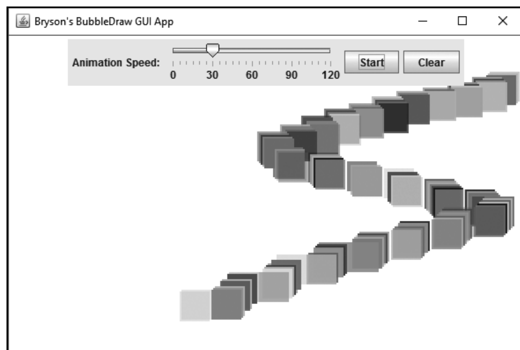


图 10-19　运行 PixelDraw 2.0 时，如果在不停止动画的情况下拖曳鼠标，将实现堆叠式 3D 效果

这里说说涉及的数学运算：为实现方块效果，需要根据方块的尺寸将屏幕划分成网格，再在每个格子内绘制方块。尝试在构造函数 Bubble()开头像下面这样设置变量 x 和 y：

```
x = (newX / newSize) * newSize + newSize/2;
y = (newY / newSize) * newSize + newSize/2;
```

在这里使用的公式中，第一部分（(newX / newSize) * newSize）根据气泡的尺寸 newSize 将屏幕划分成网格。为了让 x 和 y 坐标与 newSize×newSize 的格子对齐，需要确保它们为 newSize 的整数倍。为此，我们这样计算 x 坐标：将 newX 除以 newSize，这样得到的是一个没有小数部分的整数；然后，将这个整数乘以 newSize，从而得到一个为 newSize 的整数倍的整数。这样，气泡的 x 坐标将位于用户单击的格子的左边缘上。例如，如果 newSize 为 10，前述先除后乘运算将导致 x 为 10 的整数倍，使其与 10×10 的网格线对齐。如果我们就此止步，气泡的中心将位于网格线交点上。由于我们要让气泡位于格子内，因此添加了第二部分（+ newSize/2），将气泡移到格子内。

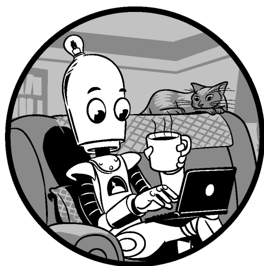
最后，为了将气泡绘制为方形而不是圆形，修改方法 draw()，在其中使用 fillRect()绘制实心矩形而不是实心椭圆。

就这么简单！但你可以（也应该）做其他的修改，进一步定制这个应用，将它变成你自己的。请在文件 BubbleDrawGUI.java 中修改 JFrame 的标题；如果你愿意，还可重构/重命名这个文件。你可随便修改，不受任何限制！

完成这些修改后，你就能绘制动画式块状画面，它们像马赛克，看起来非常漂亮。请抓取屏幕截图，并通过 Twitter 将其发送给朋友。你也可加上标签#JavaTheEasyWay，并发送给我（@brysonpayne），我将转发给几千名好友！

第 11 章

创建 Android 多点触控版 BubbleDraw 应用



本书要创建的最后一个应用是Android多点触控版BubbleDraw，它让用户能够使用 1 个或全部 10 个手指通过触摸来绘制气泡。

Android设备的处理器通常比台式机CPU小得多、慢得多。如果你遇到过“应用没有响应”错误，就见识过应用消耗过多设备处理能力的后果。有鉴于此，这里不使用定时器，而使用一种新方法——**线程化**——来实现动画，以减少应用消耗的处理能力。线程化让我们能够同时运行多个应用，这被称为**多任务**。

Andriod 版 BubbleDraw 应用还将使用**多点触控**。在图 11-1 中，气泡从多个地方喷涌而出，这是因为我的小儿子 Max 用手指触摸了这些地方。

Andriod 版 BubbleDraw 应用将重用桌面和 GUI 版的众多功能，如第 10 章编写的 `Bubble` 类的源代码。然而，在 Android 中绘制图形的方式有些不同，同时由于我们还要使用线程和多点触控，因此你需要学习一些新的应用创建技巧。你将通过创建 Android 版 BubbleDraw 应用来学习这些新技能。

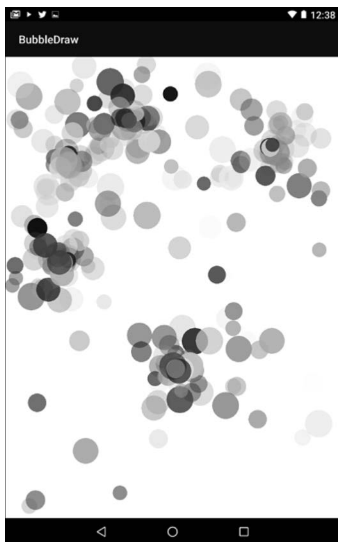


图 11-1　Android 版 BubbleDraw 应用将使用多点触控，让用户能够同时在多个地方绘制气泡

11.1　创建项目 BubbleDraw

启动 Android Studio 并关闭所有打开的项目，再单击 Start a new Android Studio project。在 Create New Project 对话框中，在文本框 Application Name 中输入 BubbleDraw，保留文本框 Company Domain 的内容（example.com 或你的网站名）不变，再单击 Next 按钮。

这次需要选择不同于以前的 API 等级。猜数游戏和应用 Secret Messages 使用的 GUI 都支持较旧的 Android 设备，但在这个应用中，需要使用方法 `drawOval()`来绘制气泡，而它要求 API 级别为 21 或更高。请将 Minimum SDK 设置为 API 21 Android 5.0（Lollipop）。

另一个不同之处是，在这个应用中，我们将使用一个空活动（而不像猜数游戏和应用 Secret Messages 那样使用基本活动），因为我们不需要基本 GUI。我们不使用常规的 GUI 应用布局，而将创建支持触摸的交互式画布。在 Add an Activity to Mobile 屏幕中，选择 Empty Activity 并单击 Next 按钮。

像创建前面的应用一样，保留文本框 Activity Name 的默认设置 `MainActivity`，但取消选择复选框 Backwards Compatibility，再单击 Finish 按钮。通过关闭向后兼容性，可让代码更简单。

与前面两个桌面版 BubbleDraw 应用一样，我们将使用两个 Java 文件，以便将与气泡相关的代码与主应用代码分开。项目打开后，如果 Project Explorer 面板不可见，单击位于屏幕左边缘的选项卡 Project 以显示它。然后，单击 Project Explorer 面板顶部的选项卡 Android。在文件夹 app▶java 中找到并右击主包 BubbleDraw（不是 androidTest 或 test 包），并选择 New▶Java Class，如图 11-2 所示。

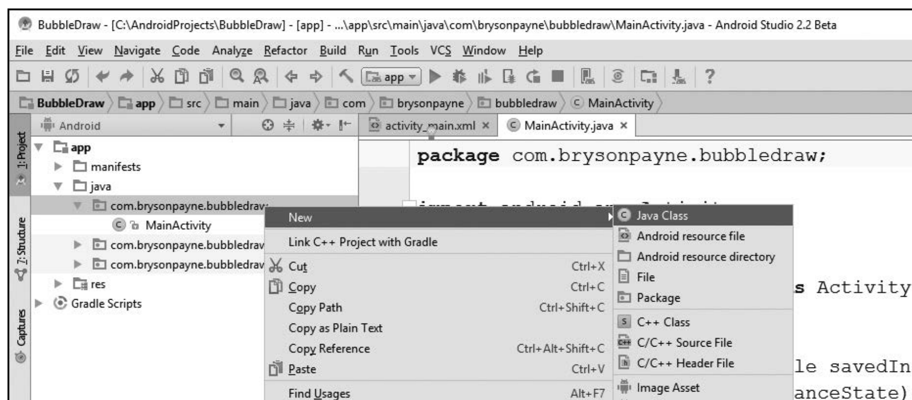


图 11-2　在主包 BubbleDraw 中添加一个 Java 类，用于放置与气泡相关的代码

在对话框 Create New Class 中，将这个类命名为 BubbleView。在 Android，View 是任何 GUI 组件。BubbleView 类的作用与桌面版中 BubblePanel 类的作用相同，所有气泡绘制代码都将放在这个类中。

Android Studio 对话框 Create New Class 让我们能够轻松地指定超类和接口。首先，我们要让 BubbleView 类继承绘图功能。为此，在文本框 Superclass 中输入 ImageView，再双击自动补全列表中的 ImageView(android.widget)，将其指定为 BubbleView 的父类。你双击后，文本框 Superclass 的内容将变成 android.widget.ImageView。

接下来，我们要实现一个 OnTouchListener 接口，让应用能够处理触摸事件——类似于我们在前两个版本中使用的鼠标事件。为此，在文本框 Interface(s)中输入 OnTouchListener，再双击自动补全列表中的 OnTouchListener(android.view.View)。你双击后，文本框 Interface(s)的内容将变成 android.view.View.OnTouchListener。

单击 OK 按钮，BubbleView 类将出现在 com.<yourdomain>.bubbledraw 包中。在 Project Explorer 中双击 BubbleView 类以编辑这个文件。类名 BubbleView 可能有红色下划线，指出其缺少代码，但接下来几节我们将编写必要的代码。

11.2　给 BubbleView 类编写代码

双击选项卡 BubbleView.java 将其扩大到占据整个屏幕，以方便编辑。我们首先在 BubbleView 类中添加一些变量，它们类似于 BubblePanel 类中的变量。与桌面和 GUI 版一样，需要一个随机数生成器和一个用于存储用户绘制的气泡的 ArrayList，还有一些用于存储默认气泡尺寸和动画延迟（单位为毫秒）的整型变量。

11.2.1　添加实现动画所需的变量

对于每个气泡，都需要指定随机的颜色和速度，因此在 BubbleView 类的左大括号后面，输

入 private Random 再双击自动补全列表中的 Random(java.util)。

在代码中添加新的对象类型时，我们都将双击相应的自动补全列表项。别忘了，Android Studio 的代码补全功能不仅可帮助你提高代码编写速度，还可避免输错或拼错类名和 import 语句。

编写好声明 private Random rand = new Random();后，再添加如下代码所示的每个变量，之后检查 import 语句，确保它们与这里显示的一致：

```
package com.yourdomain.bubbledraw; // 注意，你的包名可能不同
import android.widget.ImageView;
import android.view.View;
import java.util.ArrayList;
import java.util.Random;
public class BubbleView extends ImageView implements View.OnTouchListener {
❶ private Random rand = new Random();
❷ private ArrayList<Bubble> bubbleList;
❸ private int size = 50;
❹ private int delay = 33;
}
```

这 4 行代码类似于我们在第 9 章和第 10 章的 BubblePanel 类开头添加的变量声明，但有几个不同的地方。❶处的随机数生成器声明与旧版本中相同，它们的类型都是 java.util.Random。❷处的代码行亦如此，这里声明了一个名为 bubbleList 的 ArrayList，它也用于存储用户创建的气泡。类型说明符 Bubble 应为红色，因为我们还没有定义 Bubble 类。

在❸处，我们声明了一个整型变量，用于存储默认的气泡尺寸，但这里将默认尺寸设置得更大，因为在移动设备上像素尺寸更小。相比于台式机，手机或平板电脑的屏幕通常小得多，而像素密度高得多，因此我们将默认气泡尺寸设置为 50，让气泡更容易看清。根据你希望气泡在设备上是什么样的，你可修改这行代码，让气泡更大或更小。

在❹处，我们将帧间延迟设置为 33 毫秒，让动画速度依然为 30 fps。别忘了，为了计算动画速度，我们将 1000（毫秒）除以每秒的帧数 30（fps），得到帧间延迟 33（1000 ÷ 30）。

无论在图形还是动画方面，Android 与台式机都稍有不同，因此我们需要添加两个新变量：

```
public class BubbleView extends ImageView implements View.OnTouchListener {
    private Random rand = new Random();
    private ArrayList<Bubble> bubbleList;
    private int size = 50;
    private int delay = 33;
❶ private Paint myPaint = new Paint();
❷ private Handler h = new Handler();
}
```

❶处的代码行声明了一个名为 myPaint 的 android.graphics.Paint 对象。你可将这种对象视为用于在 Android 屏幕上绘制气泡的画笔；要在 Android 画布（Canvas）上绘制形状，必须有 Paint 对象。请在输入 Paint 后按回车键接受代码补全建议，也可在输入这行代码后单击 Paint，再按 Alt-回车键（或 Option-回车键）自动导入 android.graphics.Paint 类。

❷处的代码行声明了另一个新类型的变量——android.os.Handler 变量 h。务必导入 android.os

版 Handler 类，因为还有其他名称类似的类。这个 Handler 对象让我们能够使用线程来实现动画，你可认为它相当于桌面应用中的 Timer。然而，不同于 Timer，Handler 让我们能够与线程通信；线程是同时运行多个应用的多任务环境中的进程。Handler 不会像 Timer 那样让 CPU 在两次事件之间忙于计算，相反，它会释放 CPU，让其他任务能够运行，直到该重绘另一个动画帧。

接下来，我们来添加构造函数并开始绘制气泡。

11.2.2 创建构造函数 BubbleView()

下一步是给 BubbleView 类编写构造函数。在刚才声明的变量后面，输入如下构造函数代码：

```
public class BubbleView extends ImageView implements View.OnTouchListener {
    private Random rand = new Random();
    private ArrayList<Bubble> bubbleList;
    private int size = 50;
    private int delay = 33;
    private Paint myPaint = new Paint();
    private Handler h = new Handler();
    public BubbleView(Context context, AttributeSet attributeSet) {
        super(context, attributeSet);
    }
}
```

请使用代码自动补全功能，以导入 android.content.Context 和 android.util.AttributeSet。Android 使用这两个类来存储有关当前应用的信息，我们需要导入它们才能调用方法 super()。方法 super()调用父类 ImageView 的构造函数来设置应用和绘画屏幕。

就目前而言，只需在构造函数中添加初始化 bubbleList 的代码行：

```
public BubbleView(Context context, AttributeSet attributeSet) {
    super(context, attributeSet);
    bubbleList = new ArrayList<Bubble>();
}
```

与前两个版本一样，这里也将 bubbleList 设置为用于存储 Bubble 对象的空 ArrayList，这样用户触摸屏幕时，我们就能够将新气泡存储到 bubbleList 中。

11.2.3 准备好布局以使用 BubbleView

着手编写 BubbleView 类后，该让 GUI 布局文件在应用运行时显示 BubbleView 了。在 Project Explorer 中，打开文件夹 app▸res▸layout 中的文件 activity_main.xml，再单击窗口底部的 Text 选项卡。

使用下面的内容替换文件 activity_main.xml 的内容（将其中的包名替换为你使用的包名）：

```
<?xml version="1.0" encoding="utf-8"?>
❶ <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
❷   android:background="#000000"
❸   tools:context="com.yourdomain.bubbledraw.BubbleView">
❹   <com.yourdomain.bubbledraw.BubbleView
❺   android:layout_width="match_parent"
❻   android:layout_height="match_parent"
    />
</RelativeLayout>
```

这个应用使用默认布局 RelativeLayout❶，让你以后能够轻松地添加其他 GUI 组件。对于 RelativeLayout 的大多数属性，我们都使用了默认值，但将背景色设置成了#000000（黑色）❷。

当你将❸和❹处的包名 com.yourdomain.bubbledraw 替换你的包名时，Android 将在你替换❹处的包名时提供代码补全建议。

❹处的代码行将 BubbleView 放到布局中，而❺和❻处的代码行让程序将 BubbleView 设置成与父对象 activity_main.xml 等宽且等高。

文件 activity_main.xml 是应用运行时将加载的默认 GUI 布局视图。通过让 activity_main.xml 将 BubbleView 作为唯一的布局元素加载，并使其与布局等宽且等高，将让 BubbleView 占据整个屏幕。因此，完成这些修改后，文件 activity_main.xml 将调用 BubbleView 来显示用户绘制气泡的画布。

说到气泡，我们将重用桌面版中的 Bubble 类。

11.3　修改 Bubble 类

在 Eclipse 中打开第 10 章的项目 BubbleDrawGUI，以便访问原来的 Bubble 类。打开文件 BubblePanel.java，并滚动到其末尾定义 Bubble 类的地方。复制这个类的所有源代码——从 private class Bubble 直到倒数第二个右大括号。在这个文件中，最后一个大括号是 BubblePanel 类的右大括号，因此务必只复制方法 update()和 Bubble 类的右大括号。

复制 Bubble 类后，切换到 Android Studio，将光标放在构造函数 BubbleView()的右大括号后面，再按回车键在文件末尾的最后一个右大括号前面插入一个空行。

桌面版 Bubble 类的大多数代码都依然管用，但由于 Android 绘制图形的方式不同，我们需要修改几个地方。我们从 Bubble 类的开头附近开始。在 Android 图形中，颜色值存储为整数而不是 Color 对象，因此将 private Color color 改为 private int color，如下所示：

```
private class Bubble {
    private int x;
    private int y;
    private int size;
    private int color;
    private int xspeed, yspeed;
    private final int MAX_SPEED = 5;
```

其他变量都与以前一样。

我们还需修改构造函数 Bubble()中给变量 color 赋值的语句。为此，将其中的 new Color 替换为 Color.argb，如下所示：

```
public Bubble(int newX, int newY, int newSize) {
    x = newX;
    y = newY;
    size = newSize;
    color = Color.argb(rand.nextInt(256),
            rand.nextInt(256),
            rand.nextInt(256),
            rand.nextInt(256) );
    xspeed = rand.nextInt(MAX_SPEED * 2) - MAX_SPEED;
    yspeed = rand.nextInt(MAX_SPEED * 2) - MAX_SPEED;
}
```

务必将关键字 new 删除，因为方法 Color.argb()不创建新对象，而是将 4 个 ARGB 值（**alpha**、**红色**、**绿色**和**蓝色**）转换为一个表示颜色的整数，而这个整数可在 Android 中用来修改颜料的颜色。

这是我们首次在该应用中使用 Color 类，因此它在 Android Studio 文本编辑器中显示为红色。你可在文件开头手动添加导入语句 import android.graphics.Color;，也可单击单词 Color 并按 Alt-回车键（或 Option-回车键），让 Android Studio 为你导入这个类。按 Alt-回车键与接受代码补全建议类似，只是可在输入代码后再使用 Alt-回车键来导入类。

接下来，需要修改 Bubble 类中的整个 draw()方法。为此，请将复制而来的方法 draw()替换为如下代码：

```
        xspeed = rand.nextInt(MAX_SPEED * 2) - MAX_SPEED;
        yspeed = rand.nextInt(MAX_SPEED * 2) - MAX_SPEED;
    }
❶ public void draw(Canvas canvas) {
    ❷ myPaint.setColor(color);
    ❸ canvas.drawOval(x - size/2, y - size/2,
                x + size/2, y + size/2, myPaint);
    }
    public void update() {
```

在❶处，方法 draw()接受一个类型为 android.graphics.Canvas（而不是 java.awt.Graphics）的参数。务必导入 Canvas 类，为此可在输入代码时使用代码补全功能导入，也可在输入代码后单击 Canvas 并按 Alt-回车键（或 Option-回车键）。

在❷处，我们设置对象 myPaint 的颜色，使其使用当前气泡的颜色。

❸处的代码行有多个不同于桌面版的地方。首先，在 Android Canvas 上绘制椭圆的方法为 drawOval()，而不是 fillOval()；其次，我们使用左边缘、上边缘、右边缘、下边缘值，而不是左边缘、上边缘、宽度、高度值来指定椭圆的定界框。左边缘和上边缘值与以前相同，还是 x - size/2 和 y - size/2（别忘了，我们减去气泡宽度和高度的一半，让气泡中心位于用户在屏幕上单击的(x, y)位置处）。气泡定界框的右边缘值为 x + size/2，而下边缘值为 y + size/2，如图 11-3 所示。在桌面版中，我们指定椭圆定界框的宽度和高度，但 Android 要求指定定界框右下角

的 *x* 和 *y* 坐标，即 x + size/2 和 y + size/2。最后，方法 drawOval()要求提供一个 Paint 对象，因此我们将 myPaint 传递给它。

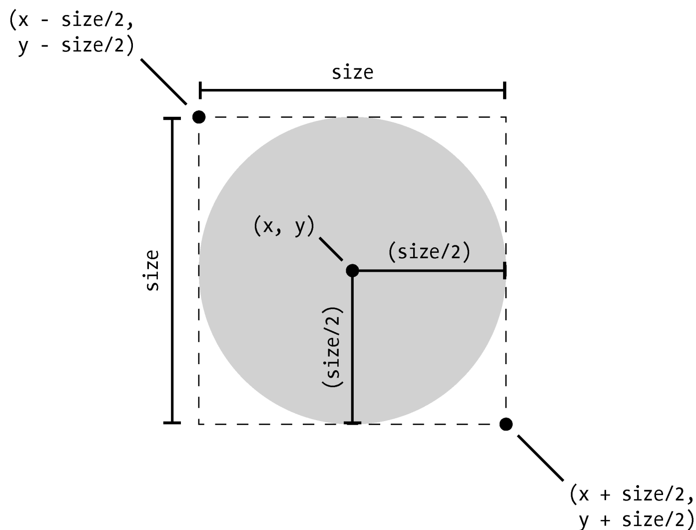


图 11-3　Android 方法 drawOval()将定界框左上角和右下角的坐标作为参数，而不像桌面工具包 Swing 那样要求提供定界框左上角的坐标以及宽度和高度

这就是将桌面版 Bubble 类移植到 Android 中需要做的全部修改。完成这些修改后，将文件存盘。接下来，我们将在屏幕上绘制所有的气泡。

11.4　使用方法 onDraw()在 Android 中绘图

我们要测试这个应用在屏幕上绘制气泡的功能，因此接下来将在 BubbleView 中添加方法 onDraw()。View 类中的方法 onDraw()类似于 JPanel 中的方法 paintComponent()：它告诉 Java 在屏幕刷新时绘制什么。

我们要绘制的是一系列气泡，因此在文件 BubbleView.java 中添加如下代码，并将其放在构造函数 BubbleView()和 Bubble 类之间：

```
public BubbleView(Context context, AttributeSet attributeSet) {
    super(context, attributeSet);
    bubbleList = new ArrayList<Bubble>();
}
❶ protected void onDraw(Canvas canvas) {
  ❷ for (Bubble b : bubbleList)
        b.draw(canvas);
}
private class Bubble {
```

在❶处，必须将方法 onDraw()声明为受保护的，并将一个 Canvas 对象作为参数，因为其签

名必须与要重写的默认方法 View.onDraw()完全相同。我们之所以需要定义这个方法，是因为所有 View 子类都必须包含它；BubbleView 是 ImageView 的子类，而 ImageView 又是 View 的子类。每当需要刷新包含 BubbleView 的屏幕时，都将调用方法 onDraw()。

在方法 onDraw()中，重用了以前的 for-each 循环，它对每个气泡调用方法 draw()❷。对于❷处的代码行，可解读为“对于 bubbleList 中的每个 Bubble 对象 b，都将其绘制到 Android Canvas 上”。

再完成两个步骤，就可测试应用绘制彩色缤纷的气泡的功能了。下面来完成这两个部分，以便对 Android 应用 BubbleDraw 进行早期 Beta 测试。

11.5 使用 100 个气泡测试 BubbleDraw

在第一个桌面版 BubbleDraw 应用中，我们编写了简短的 testBubbles()方法，它类似于下面这样：

```
public void testBubbles() {
    for(int n = 0; n < 100; n++) {
        int x = rand.nextInt(600);
        int y = rand.nextInt(400);
        int size = rand.nextInt(50);
        bubbleList.add( new Bubble(x, y, size) );
    }
    repaint();
}
```

编写方法 testBubbles()旨在确定我们能够在屏幕上绘制气泡，然后再接着实现鼠标和定时器事件处理程序。下面在 Android 版应用中做同样的事情。

11.5.1 添加方法 testBubbles()

首先，我们在文件 BubbleView.java 中添加一个稍有不同的 testBubbles()版本，并将其放在方法 onDraw()的后面：

```
protected void onDraw(Canvas canvas) {
    for (Bubble b : bubbleList)
        b.draw(canvas);
}
public void testBubbles() {
    for(int n = 0; n < 100; n++) {
     ❶ int x = rand.nextInt(600);
     ❷ int y = rand.nextInt(600);
     ❸ int s = rand.nextInt(size) + size;
     ❹ bubbleList.add( new Bubble(x, y, s) );
    }
 ❺ invalidate();
}
private class Bubble {
```

开头两行代码与 Eclipse 版 testBubbles()相同，它们声明这个方法，并创建一个执行 100 次的循环。

在❶处，我们保留 x 的取值范围不变，依然将其设置为 0~600，但如果你知道设备的分辨率，可将其设置得更大。在❷处，我们将 y（它表示气泡的垂直位置）的可能取值范围改为 0~600。

在❸处，我们生成更大的气泡：将 0 到 size 的随机数加上默认尺寸 size。这样，气泡的直径将为 50~100 像素。

在❹处，我们根据刚才生成的 3 个随机数创建一个新的 Bubble 对象，并将其添加到 bubbleList 中。

最后，在❺处，我们使用了一个以前没见过的函数——invalidate()，其作用类似于桌面版应用 BubbleDraw 使用的函数 repaint()。它告诉 Java 需要更新或刷新屏幕。函数 invalidate()清屏并调用绘制 bubbleList 中所有气泡的方法 onDraw()。

定义方法 testBubbles()后，只需在构造函数 BubbleView()中调用它来绘制气泡：

```
public BubbleView(Context context, AttributeSet attributeSet) {
    super(context, attributeSet);
    bubbleList = new ArrayList<Bubble>();
    testBubbles();
}
```

现在，应用加载时，将调用方法 testBubbles()在 bubbleList 中填充 100 个随机气泡。

11.5.2　修复 OnTouchListener 的错误

现在只需修复一个问题，就可测试应用了：类名 BubbleView 依然带红色下划线，这说明可能存在编译错误。将鼠标指向代码行 public class BubbleView，你将看到一条错误消息，指出 OnTouchListener 缺失方法 onTouch()。换而言之，这条错误消息提醒我们，BubbleView 类实现了 OnTouchListener，但我们还没有添加处理触摸事件的方法 onTouch()。

为修复这个错误，将鼠标指向并单击警告图标（红色灯泡）。在红色灯泡下方的弹出菜单中，单击 Implement methods。将出现一个弹出窗口，让你选择要实现的方法，如图 11-4 所示。

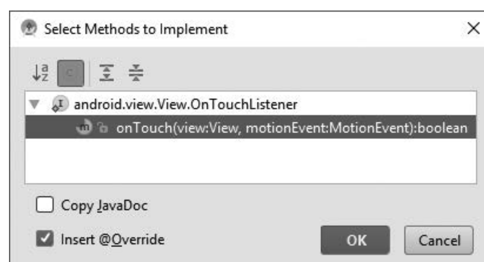


图 11-4　Android Studio 提醒你实现方法 onTouch()，确保 OnTouchListener 的完整

单击 OK 按钮，Android Studio 将在你的代码中插入方法 onTouch()以消除错误：

```
@Override
public boolean onTouch(View view, MotionEvent motionEvent) {
    return false;
}
```

如果 Android Studio 插入的代码与这里显示的不同，请将参数名改为 `view` 和 `motionEvent`。后面将修改这个方法以处理触摸事件，但我们先来测试这个应用，看看它能否绘制 100 个气泡。

11.5.3　运行应用 BubbleDraw

单击绿色 Run 按钮对应用进行测试。在 Select Deployment Target 窗口中，像第 4 章介绍的那样选择模拟器或设备。我选择的是 Nexus 6P 模拟器，如图 11-5 所示。

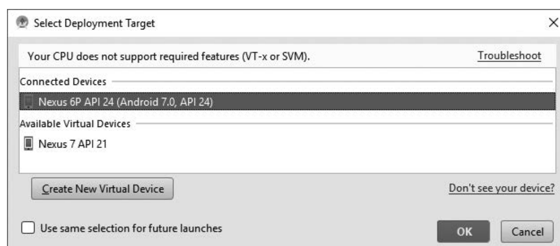


图 11-5　单击 Run 按钮编译并运行应用，再选择模拟器或设备并单击 OK 按钮

单击 OK 按钮，这将启动设备或模拟器并部署前面开发的应用 BubbleDraw。你将看到屏幕左上角充斥着气泡，如图 11-6 所示。气泡之所以出现在左上角，是因为我们生成的 x 和 y 坐标位于范围 0~600 内，而大多数 Android 设备的水平和垂直分辨率都不低于 1000 像素。别忘了，在 Java 中，坐标(0,0)位于左上角。



图 11-6　前面开发的 BubbleDraw 应用通过了测试——100 个气泡出现在屏幕左上角

与首次测试该应用的桌面版时一样，没有动画，也不会响应触摸，因为我们还没有添加这些功能。但应用能够在模拟器中运行，且在屏幕上正确地绘制了气泡。

下面来添加动画，让气泡动起来，然后，再添加触摸功能。

11.6　在 Java 中使用线程化动画和多任务

我们将在 Android 版 BubbleDraw 应用中使用线程技术来实现平滑动画，这种技术与你在其他 Java 多任务应用中使用的线程技术类似。前面说过，使用线程技术来实现动画的优点之一是，不会在等待重绘帧期间占着处理器。在其他同时运行多个线程的应用中，也可使用线程技术，如在后台查询数据库或加载文件的应用中。通过使用线程技术，可让应用在后台执行任务，避免 GUI 因等待进程执行完毕而不响应用户操作。

在手机和平板电脑等处理能力有限的设备上，线程技术显得尤其重要。如果应用响应缓慢，会惹怒用户，而线程技术可帮助避免 BubbleDraw 成为这样的应用。

本章前面创建的 Handler 对象 h 让我们能够与线程通信。在这个应用中，我们将创建一个通过更新所有气泡的位置来实现动画的线程，再使用 Handler 对象 h 来告诉这个线程何时运行。这让 Handler 对象和线程协同工作,起到与前两章的 Timer 对象一样的作用,但不会在帧间占着 CPU。

要在 Java 应用中添加线程，有两种方式：扩展 Thread 类或实现 Runnable 接口。这里将采用第二种方式。

实现了接口 Runnable()的类必须提供方法 run()，它告诉线程运行时该做什么。就应用 BubbleDraw 而言，我们要让方法 run()执行动画：移动气泡并重绘屏幕。

我们在构造函数 BubbleView()后面创建一个名为 r 的 Runnable 对象：

```
public BubbleView(Context context, AttributeSet attributeSet) {
    super(context, attributeSet);
    bubbleList = new ArrayList<Bubble>();
    testBubbles();
}
❶ private Runnable r = new Runnable() {
    @Override
  ❷ public void run() {
    }
❸ };
protected void onDraw(Canvas canvas) {
```

当你输入❶处声明的第二部分时，将出现代码补全建议，让你能够补全 new Runnable()。接受代码补全建议 java.lang.Runnable，Android Studio 将自动为你添加方法存根 public void run()❷。请注意，Runnable 对象的右大括号后面有一个分号❸，这是必不可少的，因为我们要在声明变量 r 的同时将一个新的 Runnable 对象赋给它。❸处的分号实际上是结束从❶处开始的语句。然而，自动代码补全功能不会在❸处的右大括号后面添加分号，因此务必手动添加它，以免出现编译错误。

接下来，需要在方法 run()中添加代码，告诉 Java 在线程（Runnable 对象）r 被调用时做什么：

```
private Runnable r = new Runnable() {
    @Override
    public void run() {
    ❶ for(Bubble b : bubbleList)
        ❷ b.update();
    ❸ invalidate();
    }
};
protected void onDraw(Canvas canvas) {
    for (Bubble b : bubbleList)
        b.draw(canvas);
}
```

在❶处，我们再次使用了 for-each 语句来遍历 bubbleList 中的每个 Bubble 对象 b。在这个循环的每次迭代中，我们调用 b.update()来更新每个气泡在下一个动画帧中的位置❷。

在❸处，我们在循环结束后调用函数 invalidate()，它执行清屏操作并调用方法 onDraw()来让 Java 重绘视图。

添加基于线程的动画的最后一步是，将 Handler 对象 h 与线程（Runnable 对象）r 关联起来。我们在方法 onDraw()末尾这样做：

```
protected void onDraw(Canvas canvas) {
    for (Bubble b : bubbleList)
        b.draw(canvas);
    h.postDelayed(r, delay);
}
```

方法 postDelayed()向线程 r 发送一条消息，让它 33（delay 的值）毫秒后再次运行。

请保存所做的修改并再次运行应用。你将看到 100 个测试气泡缓慢地四处飘散，充斥整个屏幕，如图 11-7 所示。

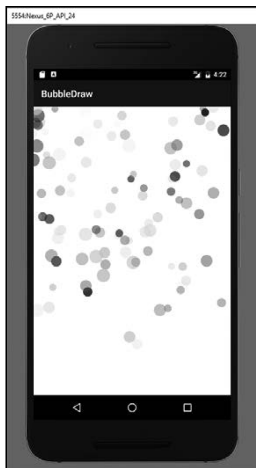


图 11-7 气泡终于动起来了，这多亏了线程化动画

虽然帧速为 30 帧每秒，但即便是速度最快的气泡，看起来也可能移动得很慢。这是因为 Android 设备或模拟器的像素密度更高。你可能还记得，我们在 Bubble 类中将 MAX_SPEED 设置成了只有每帧 5 像素，而我们模拟的 Nexus 6P 手机的屏幕分辨率为 1440 × 2560，这意味着要从屏幕一边移到另一边，即便是最快的气泡也需要 500 帧（超过 15 秒）。

下面来稍微提高点速度，将 MAX_SPEED 设置为更大的值，如 15 像素每帧：

```
private class Bubble {
    private int x;
    private int y;
    private int size;
    private int color;
    private int xspeed, yspeed;
    private final int MAX_SPEED = 15;
```

保存并再次运行应用。现在，气泡在屏幕上的移动速度更快。你可根据喜好随便调整这个速度值。

实现动画后，该添加代码让应用响应用户触摸操作了。

11.7　使用手指触摸来绘画

桌面版 BubbleDraw 应用很有趣，原因之一是用户可在任何地方单击和拖曳鼠标来绘制气泡。下面让 Android 版也这样有趣，再添加多点触控支持，将这种趣味性推到极致！

你已经知道需要在什么地方添加触摸事件处理代码——前面添加的方法 onTouch()中。

为处理触摸事件，首先需要确定用户触摸的位置，再在该处添加一个气泡。

为确定用户触摸位置的 *x* 和 *y* 坐标，可使用 motionEvent.getX()和 motionEvent.getY()。下面来给方法 onTouch()添加完整的代码，再详细解读它们：

```
public boolean onTouch(View view, MotionEvent motionEvent) {
 ❶ int x = (int) motionEvent.getX();
 ❷ int y = (int) motionEvent.getY();
 ❸ int s = rand.nextInt(size) + size;
 ❹ bubbleList.add( new Bubble(x, y, s) );
 ❺ return true;
}
```

在❶处，我们使用方法 motionEvent.getX()获取用户触摸位置的 *x* 坐标。但请注意，我们必须将返回的值转换为整数：在 Android 中，motionEvent.getX()返回一个浮点数，因此我们使用(int)进行转换。在❷处，以同样的方式获取 *y* 坐标，而在❸处，像 testBubbles()中那样生成随机尺寸，并将其存储到变量 s 中。

在❹处，我们根据给定的 x、y 和 s 值创建一个 Bubble 对象，并将其添加到 bubbleList 中。

最后一行❺需要稍作说明。请注意，方法 onTouch()返回一个布尔值，这意味着它必须返回 true 或 false。在 Android 中，如果对触摸事件做了全面处理，应让方法 onTouch()返回 true。

如果你要让 Android 去处理触摸事件（如滚动或缩放），应让事件处理程序 onTouch()返回 false。

就这个绘图应用而言，我们不想在用户滑动时让 Android 滚动屏幕：我们在用户触摸的地方添加了一个气泡，对触摸事件做了全面处理，因此返回 true。

最后一步与桌面版 BubbleDraw 应用中处理鼠标监听器的方式相同：在构造函数中添加监听器。为此，向上滚动到构造函数 BubbleView()处，将调用函数 testBubbles()的代码注释掉，并添加如下代码行：

```
public BubbleView(Context context, AttributeSet attributeSet) {
    super(context, attributeSet);
    bubbleList = new ArrayList<Bubble>();
    // testBubbles();
    setOnTouchListener(this);
}
```

我们注释掉了调用 testBubbles()的代码，因为不再需要 100 个测试气泡。我们将这样添加气泡：在 Android 设备的屏幕上触摸；在 Android 模拟器中单击并拖曳鼠标来模拟触摸。语句 setOnTouchListener(this)添加一个触摸事件监听器，让 Java 将触摸事件交给 this（一个 BubbleView 对象）去处理。

完成这些修改后，就可尝试运行应用了。保存代码并在模拟器中运行应用。在模拟器窗口中单击并拖曳鼠标以模拟在屏幕上拖曳手指。你将看到气泡从触摸的位置喷涌而出，如图 11-8 所示。

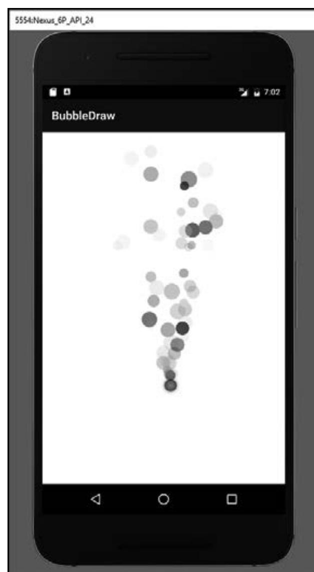


图 11-8　单击并拖曳鼠标以模拟用单个手指触摸模拟器的情形，将有一串气泡喷涌而出

你也可在 Android 设备上运行这个应用。下一节将复习如何这样做，但在此之前，我们先来添加处理多个触摸事件的功能。

11.7.1　同时使用 10 个手指进行多点触控绘画

你可能使用过支持多点触控的应用，如这样的双人对战游戏：你使用一个或多个手指控制屏幕一边的物体，而对手控制屏幕另一边的物体。如果你玩过这样的游戏，就领会到了多点触控给应用和游戏带来的强大威力。

好消息是，在 Android 中，处理多个触摸事件的代码几乎与处理单点触控的代码一样简单。事实上，我们只需在 onTouch()中添加一条语句，并修改其他两行代码，就能够让应用处理多个触摸事件。

要获悉同时发生了多少个触摸事件，可使用方法 getPointerCount()，它返回屏幕上当前有多少个**触点**（pointer，触摸事件或手指），这些触点存储在 MotionEvent 中。

我们可使用一个 for 循环在每个触点处添加一个气泡：

```
public boolean onTouch(View view, MotionEvent motionEvent) {
 ❶ for (int n = 0; n < motionEvent.getPointerCount(); n++) {
     ❷ int x = (int) motionEvent.getX(n);
     ❸ int y = (int) motionEvent.getY(n);
        int s = rand.nextInt(size) + size;
        bubbleList.add(new Bubble(x, y, s));
 ❹ }
    return true;
}
```

在❶处，我们添加了一个 for 循环，它将变量 n 从 0 递增到当前触摸事件包含的触点数。方法 motionEvent.getPointerCount()返回触点数。如果只有一个触点，getPointerCount()将返回 1，因此循环只运行一次（n = 0）。如果有两个触点，n 将依次为 0 和 1，以此类推。

在❷处，我们修改调用 motionEvent.getX()的代码——在 getX()的括号内插入 n。触摸事件的触点是经过编号的，因此通过将变量 n 作为参数传递给 motionEvent.getX()，将获取当前触摸事件中第 $n+1$ 个触点的 x 坐标。换而言之，getX(0)将返回第 1 个触点的 x 坐标，getX(1)将返回第 2 个触点的 x 坐标，以此类推。在❸处，我们做同样的处理：使用 getY(n)来获取每个触点的 y 坐标。最后，别忘了给 for 循环加上右大括号❹。

就这么简单！Java 和 Android 使得处理多个触摸事件非常容易。

11.7.2　在 Android 设备上测试多点触摸事件

保存代码以便能够再次运行应用。这次我们将在 Android 设备上运行它。遗憾的是，在 Android 模拟器中，难以使用单个鼠标来模拟多点触摸事件，因此你需要在手机或平板电脑上运行这个应用。

首先，使用 USB 电缆将 Android 设备连接到运行 Android Studio 的计算机。在设备上允许从计算机进行 USB 调试（参见 4.8 节）。如果应用 BubbleDraw 正在模拟器中运行，请将其关闭，再单击按钮 Run 或选择菜单 Run▶Run ‘app’。

在 Select Deployment Target 窗口中，找到并选择你的设备（我的设备为 Asus Nexus 7）并单

击 OK 按钮。Android Studio 将重新编译应用并将其部署到设备中。

应用启动后，显示的是一个黑色屏幕，标题栏包含“BubbleDraw”。但当你将一个或多个手指放到屏幕上后，应用将变得有趣得多，如图 11-9 所示。

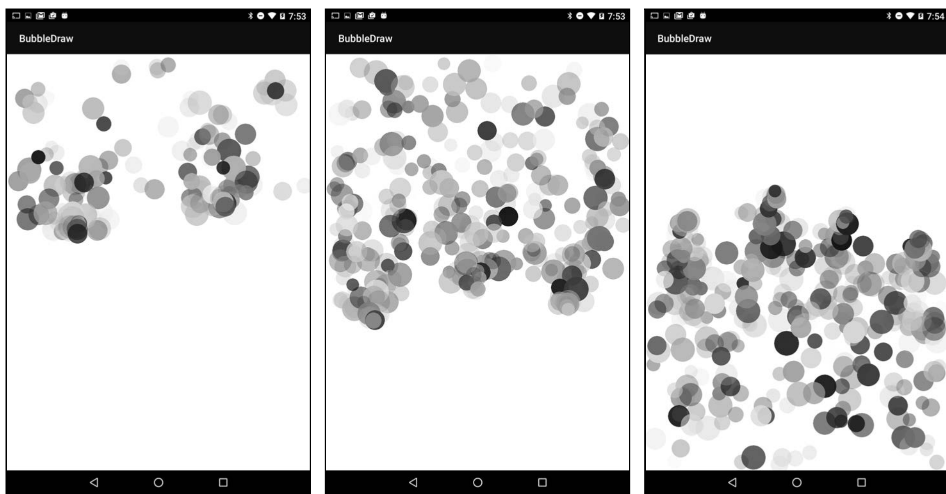


图 11-9　使用两个（左）、三个（中）和四个（右）手指触摸 Android 设备的屏幕，创建多串色彩缤纷的气泡

当你用手指在屏幕上拖曳时，将看到气泡从指尖喷涌而出，无论是使用一个、两个、五个乃至十个手指都如此。

还有一项很酷的功能我们没有介绍：要清屏，只需将设备转向一侧（务必在设备上启用自动旋转）。设备朝向发生变化时，应用将重新初始化 `BubbleView`，这将重置 `bubbleList` 并清屏。这是一种很酷的效果，让应用看起来更触手可及、更具交互性。

我们还需对应用 BubbleDraw 做一项定制：将默认的 Android 应用图标替换为自定义图标。

11.8　修改应用的启动图标

到目前为止，本书中所有的应用都使用默认的 Android 应用启动图标——绿色机器人。如果要在 BubbleDraw 应用中使用自定义图标，如公司徽标或如图 11-10 所示的应用屏幕截图，该如何做呢？

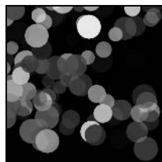


图 11-10　从应用 BubbleDraw 的屏幕截图裁剪得到的自定义图像

要给应用指定自定义图标，需要创建一个自定义的 ic_launcher.png 文件，然后将其复制到文件夹 app▸src▸main▸res▸drawable 中，再修改文件 AndroidManifest.xml，指定将这个图标用作应用启动图标。

11.8.1　创建自定义应用图标

默认情况下，Android 将应用启动图标命名为 ic_launcher.png。如果你打开文件夹 app▸src▸main▸res▸mipmap，将看到多个大小各异的 ic_launcher.png 文件，它们分别存储在文件夹 mipmap_mdpi、mipmap_xxhdpi 等中，这些文件对应于不同的手机和平板电脑的屏幕尺寸。

出于方便考虑，我们将把自定义图像也命名为 ic_launcher.png。请使用你喜欢的图像编辑程序，创建一个要用作应用启动图标的 PNG 图像（网站 http://www.gimp.org/提供了一个很不错的免费图像编辑器，https://www.pixlr.com/editor/也提供了可在线免费使用的编辑器）。最好使用方形图像，但 Android 也支持不完全是方形的图像。我使用的图像为 156 像素 × 156 像素，但 64 × 64 到 256 × 256 的任何尺寸都行。

注意　如果你要像我一样使用应用的屏幕截图，可在应用运行时截屏。这很容易，只需同时按住 Android 设备的休眠/唤醒键和降低音量键，直到屏幕闪烁表明图像已保存。在 Android 设备的“图库”应用中，找到“截屏录屏”，你将看到这个截屏图像。你可将这幅图像编辑为 256 像素 × 256 像素或更小，为此，可通过 E-mail 将其发送给自己，以便在计算机中进行编辑，也可直接在 Android 设备上进行裁剪，再通过 E-mail 发送给自己。

在图像编辑器中，将文件保存或导出为 ic_launcher.png。接下来，我们将在 Android Studio 中，将这幅图像复制并粘贴到项目 BubbleDraw 中。

11.8.2　将自定义图标添加到应用中

创建自定义应用图标文件 ic_launcher.png 后，打开它所在的文件夹并复制它。

在 Android Studio 中，在 Project Explorer 面板中展开文件夹 app▸res▸drawable 或 app▸src▸main▸res▸drawable，并将你新创建的图像 ic_launcher.png 粘贴到其中，如图 11-11 所示。

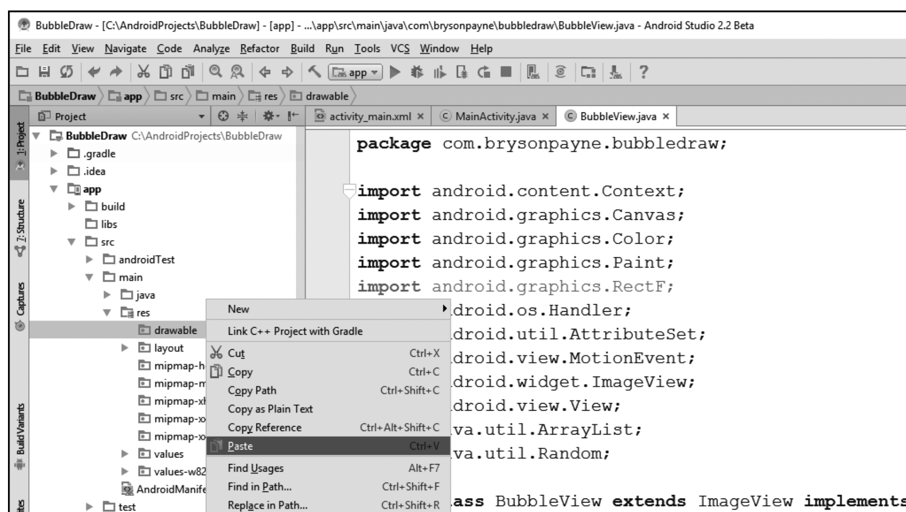


图 11-11　将新创建的自定义应用图标粘贴到文件夹 app▶src▶main▶res▶drawable 中

将出现 Copy 对话框，让你确认文件位置，请单击 OK 按钮。

将图标复制到文件夹 drawable 后，就可打开这个文件夹，再双击 ic_launcher.png 在 Android Studio 中预览它。

将新创建的图标添加到项目 BubbleDraw 中后，就可让 Android 将这幅图像用作应用图标了。

11.8.3　显示自定义图标

最后一步是编辑应用的文件 AndroidManifest.xml。文件 AndroidManifest.xml 向 Android 操作系统描述了应用的基本结构、属性和功能。

请展开文件夹 app▶src▶main（或 app▶manifests），并打开 AndroidManifest.xml。在这个文件的开头附近，找到条目 `android:icon`，并将其值改为你刚放到文件夹 drawable 中的新文件：

```
<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
```

这让 Android 在项目的文件夹 drawable 中查找图像文件 ic_launcher。在清单中，我们省略了扩展名.png。

现在，将文件 AndroidManifest.xml 存盘，并单击 Run 按钮，以使用新图标编译并部署应用。模拟器或 Android 设备中的应用 BubbleDraw 更新后，你将看到其新图标，如图 11-12 所示。



图 11-12　我们定制了应用 BubbleDraw 的启动图标

在 Android 中，我们创建的自定义 PNG 图像被用作应用的图标了！

11.8.4　修改应用名称

你还可定制出现在应用图标和标题栏中的应用名称。为此，在左边的 Project Explorer 面板中，展开文件夹 app▸src▸main▸res▸values，并打开其中的文件 strings.xml。这个文件存储了名为 `app_name` 的字符串，该字符串被用于启动图标中（如图 11-12 所示）以及应用的标题栏中。

在当前的应用名称 BubbleDraw 中，单词之间没有空格，这有点怪异，因此我们将在单词之间添加空格。可在文件 strings.xml 中定义变量 `app_name` 的 XML 代码行中这样做：

```
<resources>
    <string name="app_name">Bubble Draw</string>
</resources>
```

当然，你可以随便给应用命名，如使用 `*Your Name*\'s Bubble Draw App`，但应用图标下方只能容纳十一二个字符，因此你在主屏幕上看到的可能是 `*Your Name*'s...`。对于应用名称中的特殊字符（如单引号），必须使用反斜杠进行转义。

知道如何定制应用图标后，回过头去定制猜数游戏和应用 Secret Messages 的图标。请不断改进应用并尝试新鲜事物，这是深入学习编码的最佳方式。

11.9 小结

从第 1 章到这里，你学习了很多内容。通过创建三个完整的移动和桌面应用，你学到了大量编程技能并将其付诸实践。最重要的是，你学会了如何逐步改进应用：不断地添加功能，直到应用的行为与你期望的完全一致。

在本章中，你巩固了多项技能，并学习了如下新的编程技能：

- 在 Android 中绘制图形；
- 在 Android Studio 项目中添加新类；
- 声明变量和导入类；
- 从头开始编写类构造函数；
- 将 Java AWT 图形转换为 Android Canvas 图形；
- 使用方法 onDraw()在 ImageView 上绘图；
- 在 Java 中创建 Runnable 对象以实现线程；
- 在 Java 中使用 Handler 与线程通信；
- 使用线程改善动画效率；
- 处理多个触摸事件以及使用 MotionEvent 的方法确定触点在屏幕上的位置；
- 定制应用的启动图标和名称。

11.10 编程练习

为复习并使用学到的知识以及获得更多的编程技能，请尝试完成这里的编程练习。如果遇到困难，可从本书配套网站（https://www.nostarch.com/learnjava/）下载示例解决方案。

11.10.1 编程练习 1：区别对待单点触摸事件和多点触摸事件（1）

本章介绍了如何处理单点触摸事件和多点触摸事件。在这个编程练习中，你将区分单点触摸事件和多点触摸事件。为此，你将修改方法 onTouch()的逻辑，在用户用一个手指触摸屏幕时绘制较大的气泡，而在用户用多个手指触摸屏幕时绘制较小的气泡。

别忘了，要获悉发生了多少个触摸事件，可通过传递给方法 onTouch()的 MotionEvent 参数调用方法 getPointerCount()。

为加强练习，请修改本章应用的代码，使得用户触摸屏幕时使用的手指越多，生成的气泡越小。

11.10.2 编程练习 2：区别对待单点触摸事件和多点触摸事件（2）

完成编程练习 1 后，再尝试完成这个编程练习。修改 Bubble 类和方法 onTouch()，让用一个手指触摸时绘制的气泡的 xspeed 和 yspeed 值都相同，使它们成为一个整体，同时支持使用多个手指来绘制四处飘散的气泡。

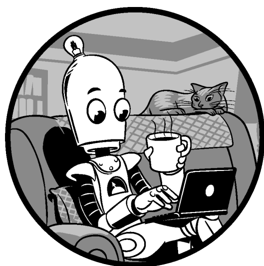
为此，需要像编程练习 1 那样修改方法 onTouch()。但为了将气泡编组，需要让有些气泡的速度是固定的，并让其他气泡的速度是随机的：对于单点触摸事件发生期间绘制的气泡，让它们都以相同的速度移动，从而看起来像是一个整体；对于多点触摸事件发生期间绘制的气泡，依然给它们指定随机的速度。

为完成这项任务，可再创建一个 Bubble()构造函数。它类似于我们为 Bubble 类创建的构造函数 Bubble(x, y, size)，但接受的参数数目不同。

例如，创建第二个 Bubble()构造函数时，可让它接受 5 个参数：x、y、size、xspeed 和 yspeed。然后，在用户使用一个手指触摸屏幕时调用这个构造函数，让所有气泡的速度都相同；在用户使用多个手指触摸屏幕时调用原来的构造函数，让气泡四处飘散。

附录 A

调试及避免常见错误



按本书的介绍编写程序时，你可能多次犯错或输入不正确，这个附录将总结几个常见且必须避开的编程陷阱。Eclipse 和 Android Studio 都提供了一个不错的功能——**语法着色**，对调试很有帮助。在 IDE 中，根据语法以不同的颜色显示类名、函数、变量类型、字符串、注释等。

语法着色可帮助你迅速发现输入错误和其他问题。例如，如果你忘记在字符串末尾加上双引号，行尾分号的颜色将与屏幕上其他地方的分号颜色不同。介绍这些常见错误时，我们将尝试在本书前面编写的应用中引入它们，并查看 Eclipse 和 Android Studio 发出的警告。不要怕破坏应用，因为随时都可修复——可使用原来的代码清单、按 Ctrl-Z（⌘-Z）或选择菜单 Edit▸Undo。

A.1 拼写和大小写

在任何编程语言中，拼写都很重要，但在 Java 中，大小写也很重要。例如，对于类型名 `Scanner` 或 `String`，如果你将其中的 `S` 小写，Eclipse 中的文本编辑器将给类名加上红色下划线，而 Android Studio 将把类名显示为红色。这看似愚蠢，但 Java 只能看懂 `String`，看不懂 `string`。

对变量名（如 `playAgain` 或 `theNumber`）来说，大小写也很重要。例如，如果我们不小心将 `p` 大写并将 `A` 小写，从而将 `playAgain` 拼写成了 `Playagain`，Java 将不会认为我们指的是变量 `playAgain`。下面来看看 IDE 是如何帮助我们找到并修复这种拼写错误的，先看 Eclipse，再看 Android Studio。

A.1.1 在 Eclipse 中校正输入错误

发现错误时，IDE 会提醒我们：Eclipse 给拼写不正确的单词加上红色下划线，而 Android Studio 将单词显示为红色。图 A-1 显示了 Eclipse 如何突出我在第 2 章的猜数游戏中引入的两个错误。

```
system cannot be resolved    system.out.println("Would you like to
                             playagain = scan.next();
```

图 A-1　Eclipse 可帮助我们发现拼写错误和大小写错误

在这里，我在 system 中使用了小写 s，Eclipse 给这个有拼写错误的单词加上了下划线，并在屏幕左边添加了错误警告，你可将鼠标指向错误警告来查看错误消息，这里显示的是 system cannot be resolved。这意味着 Java 不知道 system 是什么，因为它只能识别使用大写 S 的 System。对于下一行中的 playagain，Eclipse 也这样处理——因为它应该是 playAgain。

别忘了，你可使用 Eclipse 的内容助手功能来修复众多类似这样的错误。将鼠标指向图 A-1 所示的拼写不正确的单词，将出现类似于图 A-2 的内容助手菜单。

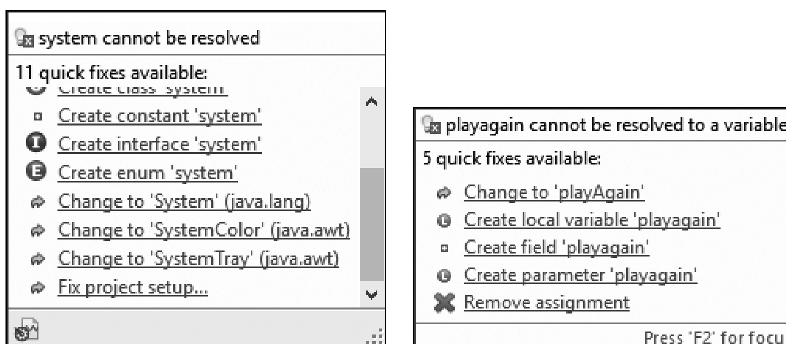


图 A-2　Eclipse 的内容助手功能提供有关错误的信息和可能的修复方案

对于第一个拼写错误，Eclipse 不仅指出了它无法解析（理解）system，还提供了多种校正方案，其中包括 Change to 'System'(java.lang)（图 A-2 中的倒数第 4 个）。内容助手功能并非总能提供正确答案，但在这里，它提供的 11 个答案中有一个是正确的：改为 System。对于第二个拼写错误，Eclipse 提供的第一个快速修复方案是正确的：Change to 'playAgain'。如果 Eclipse 的内容助手提供了正确的修复方案，可单击它，拼写错误和大小写不正确的代码将被替换为正确的代码。

A.1.2　在 Android Studio 中校正输入错误

Android Studio 将有疑问的地方用红色显示，以帮助我们发现错误。在图 A-3 所示的示例中，我故意拼错了关键字 public，将其写成了 Public。

```
45        }
46        Public void newGame() {
47   Cannot resolve symbol 'Public'  mber = (int)(Math.random() * 100 + 1);
48        }
49
```

图 A-3　Android Studio 提醒我们有个关键字的大小写不正确

注意，错误消息类似于前面在 Eclipse 中看到的：Java 指出它无法解析（理解）符号 Public。然而，在修复错误方面，Android Studio 提供的支持力度可能没有 Eclipse 大，如图 A-4 所示。

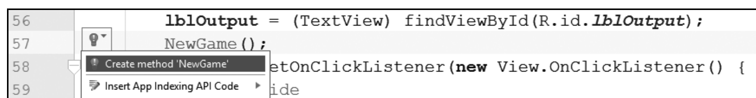


图 A-4　在 Android Studio 中，按 Alt-回车键（或 Option-回车键）可显示快速修复列表，但这里没有提供正确的修复方案

在图 A-4 中，我将 newGame()错误地拼写成了 NewGame()——将字母 n 大写了。Android Studio 正确地将它显示成了红色，让我知道存在错误，但当我单击这些代码并按 Alt-回车键以显示快速修复列表时，结果如何呢？Android Studio 未能指出正确的拼写为 newGame，而是让我再创建一个名为 NewGame 的方法。对于这个输入错误的方法名，这不是正确的修复方案，因此你需要前往定义方法 public void newGame()的地方确定正确的拼写，再手动修复输入错误。

Android Studio 和 Eclipse 都会竭尽全力地帮助你找出错误，且通常会提供可能的快速修复方案，但如果你认识到了拼写和大小写错误是常见的问题，将有助于你避免它们并在它们出现后快速修复。

A.1.3　避免其他常见的拼写错误

错误地输入了事件处理程序或其他重写方法的名称时，代码调试起来将非常困难。这样的输入错误很多，从将 onCreate()和 onDraw()错误地输入为 OnCreate()和 OnDraw()，到在按钮的 ActionListener 中错误地输入 public void actionPerformed(ActionEvent e)，再到遗漏了事件处理程序 mousePressed()、mouseClicked()或 mouseDragged()中的 ed。

遵守编程约定有助于避免这些错误。你可能忘记将方法或变量名的第一个字母小写，或忘记将类名的第一个字母大写，这虽然不是错误，但在 Java 中是糟糕的做法。如果你一开始就养成了遵循 Java 约定的习惯，就可在以后节省时间并避免麻烦。

A.2　与比较相关的错误

别忘了，两个等号（==）是比较运算符，表示“等于”。可别将其与赋值运算符（=）混为一谈。例如，要将 5 赋给变量 number，应像下面这样做：

```
int number = 5; // 将 5 赋给变量 number
```

而要比较变量的值，应使用两个等号：

```
if (number == 5) // 如果 number“等于”5
```

另外，别忘了检查字符串时，应使用方法 equals()，而不是运算符==。例如，条件

if(playAgain == "y")在任何情况下都将为 false。正确的 if 语句如下：

```
if (playAgain.equals("y"))
```

String 类的方法 equals()检查两个字符串的**内容**是否相同，这通常是我们比较字符串的目的所在。比较两个对象时，也使用方法 equals()，例如，if(bubble1.equals(bubble2))检查两个 Bubble 变量是否指向同一个 Bubble 对象。

A.3　编组符号

编写 Java 代码时，确保括号、尖括号和大括号成对出现也很重要。我们将这些括号称为**编组符号**（grouping symbols），因为它们将其他编程元素编组，而你绝不能让左编组符号没有配套的右编组符号。Eclipse 和 Android Studio 都提供了各种途径，可用于发现并修复没有配套右编组符号的问题。

A.3.1　在 Eclipse 中修复编组符号错误

Eclipse 提供了两种修复编组符号错误的途径。如果你在条件或函数中遗漏了括号，Eclipse 将给离遗漏的括号最近的单词加上红色下划线，以指出这种错误，如图 A-5 所示。

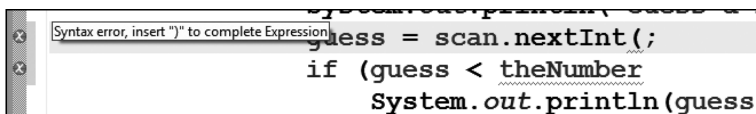


图 A-5　Eclipse 帮助你发现与缺失括号相关的错误，有时还会提供修复方案

同样，Eclipse 会尽可能在你输入代码时帮助你发现错误。在图 A-5 中，我在两行末尾都遗漏了括号，而 Eclipse 使用红色错误符号指出了这一点。如果你单击编辑器左边缘上针对第一行的红色错误符号，将显示修复建议，让你插入右括号。

如果将光标放在左大括号或右大括号旁边，Eclipse 将给与之配套的大括号加上轮廓。请在程序中的任何一个大括号旁边单击，看看结果是什么样的。

另外，别忘了 Eclipse 能够替我们正确地缩进，为此只需选择要缩进的代码，再按 Ctrl-I。对于较短的程序，这也许只是锦上添花，但对于横跨多页的大程序，这就是雪中送炭，可极大地帮助避免大括号缺失或位置不对带来的错误。

A.3.2　Android Studio 的代码补全功能

在发现编组符号缺失方面，Android Studio 提供的帮助比 Eclipse 还要大。Android Studio 的代码补全功能能够自动添加左右括号、尖括号和大括号。

我们来看一个例子。请打开第 4 章的项目 GuessingGame，并在方法 onCreate()中找到调用

方法 newGame()的代码行：

```
lblOutput = (TextView) findViewById(R.id.lblOutput);
newGame();
btnGuess.setOnClickListener(new View.OnClickListener() {
```

将行尾的括号和分号都删除：

```
lblOutput = (TextView) findViewById(R.id.lblOutput);
newGame
btnGuess.setOnClickListener(new View.OnClickListener() {
```

Android Studio 将把 newGame 显示为红色，让你知道这里存在错误。在这种情况下，你可使用代码补全功能来添加缺失的括号和分号。为自动补全这条语句，将光标放在 newGame 后面（如图 A-6 所示），再按 Ctrl-空格执行基本的代码补全。

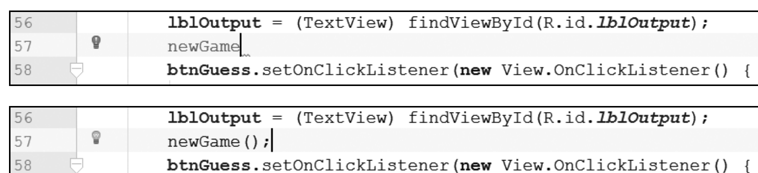


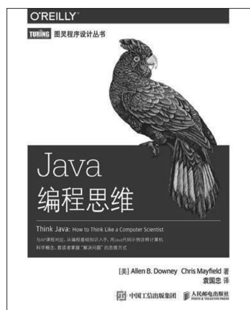
图 A-6 将光标放在缺失括号的代码行行尾（上），并按 Ctrl-空格使用 Android Studio 代码补全功能补全语句（下）

注意，代码补全功能在行尾添加了缺失的括号和分号。在 Android Studio 中，有三个代码补全快捷键，组合键 Ctrl-空格只是其中的第一个。第二个是 Ctrl-Shift-空格，它在一个弹出窗口中显示相关的选项，被称为**智能补全**；再次按这个键盘快捷键可展开代码补全选项列表。最后，**语句补全**（在 Windows 和 Linux 系统中，为 Ctrl-Shift-回车键；在 macOS 系统中为⌘-Shift-回车键）将添加右括号、右尖括号或右大括号，并在必要时添加分号。为尝试使用语句补全，可删除方法、if 语句或 for 循环的右括号，再按 Ctrl-Shift-回车键。Android Studio 的语句补全功能通常会添加缺失的配套编组符号，并在必要时添加分号，让你能够更快速、更轻松地编写代码，还有助于避免错误。

A.4 小结

使用 Java 编程时，你将遇到的错误远不止这些，但如你所见，Eclipse 和 Android 提供了对初学者和经验丰富的专家都极有帮助的工具。随着你使用 Java 编写的代码越来越多，你发现并校正错误的能力也将越来越强，但你可能永远无法避免代码出现错误。我从事编程工作 30 多年了，编写的代码依然需要调试。为了避免、发现和修复代码中的错误，应从一开始就学习良好的编程实践，并充分利用 Eclipse 和 Android Studio 等专业工具提供的支持，这至关重要。

技术改变世界 · 阅读塑造人生



Java 编程思维

◆ 与AP课程对应，从编程基础知识入手，用Java代码示例诠释计算机科学概念，教读者掌握“解决问题”的思维方式。

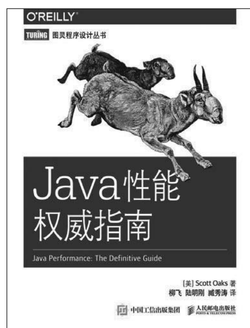
作者： Allen B. Downey Chris Mayfield
译者： 袁国忠



Java 测试驱动开发

◆ Java开发必读！
◆ 从使用TDD开始，改善设计和代码的质量、简化重构工作、提高代码覆盖率。

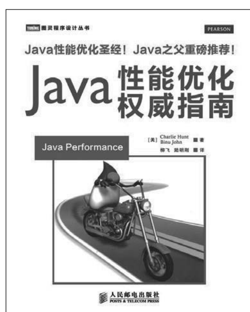
作者： Viktor Farcic Alex Garcia
译者： 袁国忠



Java 性能权威指南

◆ 深入理解Java平台性能，让你的程序如虎添翼！

作者： Scott Oaks
译者： 柳飞 陆明刚 臧秀涛



Java 性能优化权威指南

◆ Java性能优化圣经！Java之父重磅推荐！

作者： Charlie Hunt Binu John
译者： 柳飞 陆明刚



微信连接



回复“Java”查看相关书单



微博连接

关注@图灵教育 每日分享IT好书

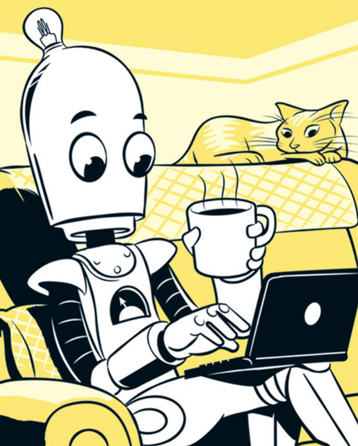


QQ连接

图灵读者官方群I：218139230
图灵读者官方群II：164939616

图灵社区
iTuring.cn

在线出版，电子书，《码农》杂志，图灵访谈



亚马逊读者评论

“许多计算机图书的作者要么是没有实战经验的教授，要么是没有教学经验的程序员，而本书的作者显然既有实战经验又有教学经验，了解如何学习才能更轻松地打开编程世界的大门。我打算和我的孩子们一起攻读这本书。”

“如果你还没学过Java，建议选择本书作为入门指南。书中内容简单易懂，对初学者非常友好，在学习使用Java创建游戏和应用的过程中，你的自信心也会逐渐增强。”

“这本书并没有对Java这门强大的语言进行全方位的介绍，但内容经过精心编排和设计，足以帮助初学者打下坚实的基础。”

Java是世界上非常流行的一门功能强大的多平台编程语言，但学习曲线十分陡峭。本书从实际项目着手，针对Java零基础读者，从安装和设置Java开发工具开始，一步步帮你消除Java学习中的拦路虎，让你能够立即开发真实可行的应用程序。

在创建应用的过程中，你将学习：

- 执行计算、操作文本字符串以及生成随机颜色；
- 使用条件、循环和方法让程序简洁且响应迅速；
- 创建函数以重用代码并节省时间；
- 创建图形用户界面元素，包括按钮、菜单、弹出框和滑条；
- 利用Eclipse和Android Studio提供的功能调试代码，发现、修复和避免常见错误。



图灵社区：iTuring.cn
热线：(010)51095186转600

分类建议 计算机/Java

人民邮电出版社网址：www.ptpress.com.cn



ISBN 978-7-115-48219-8
定价：59.00元

看完了

如果您对本书内容有疑问，可发邮件至 contact@turingbook.com，会有编辑或作译者协助答疑。也可访问图灵社区，参与本书讨论。

如果是有关电子书的建议或问题，请联系专用客服邮箱: ebook@turingbook.com。

在这可以找到我们:

微博 @图灵教育：好书、活动每日播报
微博 @图灵社区：电子书和好文章的消息
微博 @图灵新知：图灵教育的科普小组
微信 图灵访谈：ituring_interview，讲述码农精彩人生
微信 图灵教育：turingbooks